Three Essays in Computational Economics

Dissertation
submitted to the Faculty of Economics,
Business Administration and Information Technology
of the University of Zurich

to obtain the degree of
Doktor der Wirtschaftswissenschaften, Dr. oec.
(corresponds to Doctor of Philosophy, PhD)

presented by

Gregor Reich
from Germany

approved in February 2015 at the request of
Prof. Karl Schmedders, PhD
Prof. Che-Lin Su, PhD

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, 11.2.2015

Chairman of the Doctoral Board: Prof. Dr. Josef Zweimüller

Abstract

This thesis compiles three papers on different topics in computational economics: First, we present a new method to recursively integrate expectations over serially correlated latent variables with continuous support in maximum likelihood estimations, using highly efficient quadrature rules and interpolation; we apply the method to the dynamic discrete choice model of Rust (1987). Second, we present a method to use the constrained optimization approach to the estimation of dynamic models (MPEC; Su and Judd, 2012) in conjunction with grid adaption for state variables with continuous support; we use grid adaption by node movement, and derive sufficient optimality conditions for the underlying function approximation from the equioscillation theorem, which enables us to solve the estimation and the approximation problem simultaneously. Third, we present a simulation study using agent-based modelling to investigate whether a model of monopolistic competition can reach its equilibrium without common knowledge of aggregate demand and sophisticated utility optimization capabilities of the agents.

To Carola

Acknowledgements

I am heavily indebted and very grateful to my supervisors Karl Schmedders and Che-Lin Su, as well as to Ken Judd, for their teaching, guidance and support. Less teaching, but as much support and sympathy came from Carola, who always backed me up, especially when difficult decisions needed to be made. I thank my parents, grandparents, and my close friends for their continuous support and encouragement during this project. Finally, I thank my co-author Ole Wilms for the great co-operation.

Contents

Part I

Introduction

Introduction ("Rahmenpapier")

Many modern economic models exhibit a degree of complexity that makes an analytic solution intractable. If the assumptions and simplifications that induce closed form expressions are too restrictive, or simply do not exist, researchers have to apply methods from numerical analysis and computer science to obtain approximate solutions to their problems; thus, the term "computational economics" subsumes the development and application of numerical solution methods in the different fields of economic research, such as econometrics, finance, macro-, and microeconomics.

Not having to restrict models by simplifying assumptions in order to make them analytically tractable certainly offers many more degrees of freedom. In all papers of this thesis, and in the paper Larsen, Oswald, Reich, and Wunderli (2012) which is the result of a joint student project at the "Institute on Computational Economics" at University of Chicago in 2010, we generalize existing models and methods by relaxing restrictive assumptions, that were originally imposed for easier computation. However, using an analogy from statistics, while including more variables in a linear regression will always increase its coefficient of determination, R^2, it will potentially also increase the variance of the estimator, if multicollinearity arises. Analogously, allowing for more general specifications in economic models usually comes at a price, such as higher complexity of the solution procedure, or a lower precision (or higher variance) of the solution itself. Consequently, all papers of this thesis rigorously test the significance of improvement towards the original, more constraint specifications.

The thesis contributes the following three papers:

Chapter 1 addresses a widely used but restrictive assumption for the estimation of dynamic discrete choice models (DDCM): In DDCMs, decisions of agents that behave dynamically optimal are observed, along with other state variables that enter the agents' decision problems, and parameters of the underlying structural model of utility and state motion are estimated (citations are given in respective chapters). Since the estimation of the parameters together with the solution of the dynamic problem can be a difficult task computationally, a series of rather restrictive assumptions on some of the state variables — namely those that are unobservable to the econometrician and thus are not part of the data — have been made in order to reduce the computational burden by implying partial closed form solutions; among these assumptions are extreme value type I ($EV1$) iid. distributed and conditionally independent

unobserved state variables. While in Larsen et al. (2012) we test the significance of relaxing the $EV1$ assumption using numerical quadrature to approximate integrals for which otherwise closed form solutions exist, Chapter 1 takes this approach one step further to allow for serially correlated unobserved state variables.

If the unobserved state variables of a DDCM that is estimated using maximum likelihood are serially correlated, two problems arise: First, in order to solve the dynamic problem of the agent, the expected discounted continuation value has to be computed; this includes integrating over all possible future values of the state variables. For the case that the unobserved state variables are distributed $EV1$ iid. — implying serial independence — Rust (1987) derived closed form solutions for these integrals that can easily be evaluated. However, if the serial independence assumption is relaxed, no such closed form expressions exist, and thus the expectation over the value function needs to be approximated using numerical quadrature. Moreover, since the expectation of the discounted continuation value is conditional on current realizations of the states, the dimensionality of the expected value function increases, as it has to be approximated as a function of the previous period's state variables explicitly. Second, when computing a likelihood function, expectations over the unobserved state variables have to be computed as well. In contrast to the dynamic model, where solving the Bellman equation only requires the integration of the one period ahead expectation, the computation of the likelihood function requires the expectation over the full time horizon. Consequently, the dimension of the integration is proportional to the time horizon, which is usually large.

Chapter 1 addresses these issues as follows: For the approximation of the expected value function, we use Gaussian quadrature and piecewise-linear approximation paired with an iterative grid adaption scheme that makes interpolation node placement dependent on local approximation errors. The resulting collocation system is solved using a quasi-Newton method. For the integration of the likelihood function, a new algorithm is developed, the recursive likelihood function integration (RLI): Re-writing the likelihood function as a recurrence relation, it can be approximated by repeatedly applying numerical quadrature for the one step ahead expectation, and interpolation. We show that the integral over the unobserved state variables — which has previously been considered infeasible — is computed with complexity that is linear in the time horizon. The method is then applied to the well-known bus engine replacement model of Rust (1987) which is estimated using a nested fixed point algorithm coupled with value function grid adaption, and verified in an extensive Monte Carlo study. For the original data set, we find no significant serial correlation if fixed effects are controlled for, which is compatible with the findings of Rust (1987).

Chapter 2 is a methodological contribution to the estimation of dynamic models using constrained optimization approaches. Estimating the structural parameters of dynamic models such as in the DDCM case of Chapter 1 often requires the simultaneous solution of an estimation problem, for example likelihood maximization or least squared minimization, and the dynamic model. Well-known algorithms are the nested fixed point (NFXP) algorithm of Rust (1987) which completely solves the model for every "guess" of the parameter vec-

tor in the estimation process, or the mathematical programming with equilibrium constraints (MPEC) approach of Su and Judd (2012), which adds equilibrium conditions of the model as (in)equalities to the estimation problem and solves it using constraint optimization techniques. Since the constraints are usually only enforced to hold at the solution but not throughout the whole optimization, the underlying dynamic model is not completely solved in every iteration of the estimation process, which can significantly reduce computation times.

If the state variables of the underlying dynamic problem have continuous support, interpolation (or collocation) based approximation methods require the user to specify a grid of nodes at which the residual is enforced to be zero. While interpolation is very intuitive and relatively easy to implement, it is not overly rigorous in terms of approximation error control, as no inherent mechanism ensures that errors are small also between the nodes. Consequently, iterative procedures have been developed to insert new nodes in regions where the approximation error is high, in an attempt to drive down the overall approximation error. When estimating DDCM using NFXP, iterative grid refinement can efficiently be integrated as we show in Chapter 1. However, if MPEC is used, the function approximation scheme is "hard-coded" into the optimization problem: in the case of interpolation (or collocation), each interpolation node is represented by (at least) one constraint. Thus, grid refinement by node insertion cannot be used in conjunction with MPEC, as it would change the underlying optimization problem while the solver runs.

Chapter 2 therefore develops a different methodology to use grid adaption within the MPEC estimation algorithm. Instead of inserting new nodes in regions of high approximation error, nodes are moved there from regions of low error, a technique called r-adaption. However, finding node positions is a potentially difficult problem, as node movement effects not only the approximation error in the region where the node is moved to, but also where it has been moved away. Consequently, we specify the node movement problem as a generic approximation error minimization problem, where the node positions are decision variables. In order to integrate the node movement problem into the estimation with MPEC, we derive sufficient optimality conditions from the equioscillation theorem. Since these conditions form a system of (in)equalities, they can well be integrated with MPEC, such that the estimation and the approximation error minimization problem are solved simultaneously. We apply our method to the well-known bus engine replacement model of Rust (1987), which we modify to feature a continuous mileage state. Our results suggest that using flexible grids can increase accuracy of the estimation by an order of magnitude, or, if a fixed level of accuracy is targeted, it is substantially faster than using a uniform grid that yields approximately the same accuracy.

Chapter 3 uses an agent-based simulation framework to investigate to which extend commonly made assumptions about consumers' ability to explicitly optimize their utility, and the knowledge of aggregate demand function for profit maximizing producers are crucial for a market to find an equilibrium. In agent-based modelling (ABM), the smallest acting entities of an environment are modelled, and the outcome of the environment as a whole is analyzed by simulating the interactions of the single agents. The ABM paradigm gives a bottom-up per-

spective on social and economic systems, which are regarded as the complex outcome of many single interactions, potentially combined with structure imposed on higher levels. The fact that these systems are usually simulated rather than explicitly solved for an equilibrium gives the researcher a high degree of freedom on the choice of behavioural assumptions imposed on the agents; these range from simple zero-intelligence agents to highly sophisticated applications of artificial intelligence techniques. On the other hand, the analysis of outcomes of ABM simulations is often less rigorous, as mostly statistical methods are used to infer insights from the generated data.

Therefore, in Chapter 3 we ask the question whether a market of monopolistic competition can reach its equilibrium predicted from theory, without assuming that (i) consumers explicitly know their own preferences and are able solve an optimization problem to obtain their consumption decisions, and (ii) producers know the consumers' aggregate demand as a function of prices. Rather, in our model we assume that the agents gradually learn their preferences from past consumption experience, using a technique called reinforcement learning (RL). In RL, realized utility from past consumption is added up for each good, from which decision probabilities are derived. At the same time, producers are assumed to have no prior knowledge of aggregate demand, but test different price and quantity configurations in order to estimate an approximation of the demand curve to finally maximize their profits from production. We find that in a simple two goods market with sufficiently many consumers, if agents are given enough time for price adjustment and learning, the market can indeed implement the prices and quantities predicted by the analytical model of monopolistic competition. This paper has been published as Reich (2011).

Part II

Three Essays in Computational Economics

Essay 1

Recursive Likelihood Function Integration

Divide and Conquer: Recursive Likelihood Function Integration for Dynamic Discrete Choice Models with Serially Correlated Unobserved State Variables

Gregor Reich[1]
Dept. of Business Administration
University of Zurich
Moussonstrasse 15
8044 Zurich, Switzerland
gregor.reich@business.uzh.ch

November 2014

Abstract

This paper develops a method to efficiently estimate dynamic discrete choice models with serially correlated unobserved state variables such as $AR(n)$ errors. First, we solve the dynamic problem using Gaussian quadrature and interpolation over adaptively refined grids. Second, to evaluate the likelihood function, we decompose the integral over the unobserved state variables such that it can be computed recursively to high accuracy using Gaussian quadrature and interpolation. We apply this method to the bus engine replacement model of Rust [*Econometrica*, 55 (5): 999–1033, (1987)]; additionally, we verify the algorithm in a Monte Carlo study with simulated datasets.

Keywords: Dynamic discrete choice models; Numerical dynamic programming; Gaussian quadrature; Interpolation; Adaptive grids; Recursion.
JEL Classification: C25; C63.

[1]I am heavily indebted to my advisor Karl Schmedders as well as to Ken Judd, John Rust, and Che-Lin Su for their support and guidance in this project. I would also like to thank Greg Crawford, Philipp Eisenhauer, Katharina Erhardt, Dennis Kristensen, János Mayer, Bertel Schjerning, Ole Wilms, and seminar audiences at the University of Chicago, Hoover Institution, University of Zurich, and the 68th European Meeting of the Econometric Society for helpful comments and suggestions. Finally, I thank Dave Brooks for editorial comments on the manuscript.

1.1 Introduction

This paper develops a new approach to efficiently estimating dynamic discrete choice models with serially correlated unobserved state variables such as $AR(n)$ errors: First, we show how to combine some well-known methods from the literature, such as Gaussian quadrature, adaptive grid refinement, and methods for large sparse non-linear systems of equations, in order to efficiently approximate the solution to the dynamic optimization problem of the agent. Second, we develop a method to decompose and approximate the integral over the unobserved state variables that appears in the likelihood function, which has previously been considered infeasible for approximation by highly efficient deterministic integration schemes such as Gaussian quadrature; we call this procedure recursive likelihood function integration (RLI). Finally, we apply the method to the well known bus engine replacement model of Rust (1987) to estimate its parameters in the presence of serially correlated errors by using maximum likelihood, using a nested fixed point algorithm. We first apply the method to artificial data sets in order to verify the algorithm's ability to recover the parameters of the model. Then, we estimate the model using the original data set, and we find significant serial correlation for some of the subsamples of the original dataset.

Dynamic discrete choice models (DDCMs) have become a popular instrument for the econometric analysis of decision making: First, many (individual) economic decisions we actually can observe are in fact discrete in nature, for example the choice of a brand or medical treatment. Second, the underlying utility maximization problem of the agents is often dynamic in nature: decisions made today not only influence today's payoffs, rather they also influence future decisions and payoffs. By capturing these key facts, DDCMs have a wide range of uses; for recent surveys see, for example, Aguirregabiria and Mira (2010) and Keane, Todd, and Wolpin (2011).

The majority of contributions to the literature on the estimation of DDCMs make strong distributional assumptions about the errors, or, used synonymously, the unobserved state variables. Probably the most prominent example is extreme value type I $EV1$ iid distributed errors; obviously implied by the $EV1$ iid assumption, but usually stated explicitly by a conditional independence assumption (CI), the errors are assumed to be serially uncorrelated.

However, there exists a wide consensus that these assumptions are not made based on the existence of much empirical evidence, but rather for numerical tractability: $EV1$ iid errors and CI often induce closed form solutions to potentially high dimensional integrals that arise in the solution to the dynamic optimization problem and in the choice probabilities in the likelihood function. These closed form solutions go back to the work of McFadden (1974, 1981) and Rust (1987).

While relaxing the $EV1$ assumption has attracted some attention — for example, Larsen et al. (2012) test the statistical significance of allowing for more general distributions in the Rust (1987) model — several papers have developed integrated methods to estimate models without

the CI assumption, thus allowing for a general notion of serially correlated unobserved state variables. Among those are (listed alphabetically) the expectation–maximization algorithm based on conditional choice probability estimation of Arcidiacono and Miller (2011), the particle filter method of Blevins (2011), the simulation and interpolation method of Keane and Wolpin (1994), the Markov chain Monte Carlo approaches of Norets (2009, 2012), and the application of Gaussian quadrature and interpolation as discussed in Stinebrickner (2000).

While the approaches to DDCM estimation with serial correlation are diverse, most of them share a common challenge:

> "the likelihood function for a DDCM can be thought of as an integral over latent variables (the unobserved state variables). If the unobservables are serially correlated, computing this integral is very hard." (Norets, 2009)

This conclusion follows from the fact that the integral over serially correlated errors really has dimensionality proportional to the time horizon of the data, which itself can be arbitrarily large; moreover, no closed form solution for this integral exists in general.

A popular numerical approach to high dimensional integration is Monte Carlo integration (MC), because its approximation error is independent of the dimensionality of the integral. However, the approximation error usually decreases only very slowly as the number of integration nodes is increased: in order to reduce the estimated error by one order of magnitude, one usually has to increase the number of nodes by two orders of magnitude.[2] Consequently, MC is a natural choice for high dimensional integrals, but only if the integral has no structure that could potentially be exploited by more efficient methods. In contrast to MC, many quadrature rules exist that have much faster decaying errors but usually inefficient (in the worst case exponential) scaling in the dimensionality; a popular example is that of Gaussian quadrature rules, extended for multiple dimensions by the product rule.

The approach followed in this paper is to identify and exploit the structure that is present in the integral over the unobserved state variables in the likelihood function: given the serial dependence of the unobserved state variables is Markov, as, for example, with $AR(n)$ errors, the time structure allows us to decompose the high dimensional integral over the time horizon, and rewrite it as a sequence of low dimensional integrals. Then, we can approximate this sequence to high accuracy, using highly efficient approximation schemes for low dimensional integrals, including Gaussian quadrature. We show that the computational complexity of computing this integral is linear in the time horizon.

In order to evaluate the likelihood function, we need to compute the solution to the dynamic optimization problem of the agent, namely the expected value as a function of the state variables. In the presence of serial correlation, approximating the solution to the dynamic problem

[2] Formally, the variance of the Monte Carlo estimate of an integral I is proportional to $n^{-\frac{1}{2}}$, where n is the number of integration nodes.

involves different numerical tasks: First, taking the expectation of the value function is an integration over the unobserved state variables, for which, in contrast to the $EV1$ iid case, no closed form solution exists. Consequently, we have to approximate these integrals numerically, and we discuss how to apply Gaussian quadrature, which was first proposed and successfully implemented in the context of DDCMs by Stinebrickner (2000). Second, we have to approximate the expected value function as a continuous function of the unobserved state variables; in the $EV1$ iid case, this step was not necessary, because the unobserved state variables are integrated out in the closed form solution. Different approaches to value function approximation have been proposed (see, for example, Cai and Judd, 2013; Judd, 1998; Rust, 1996), and to stay flexible and generic we use interpolation over an adaptively refined grid, as proposed by Grüne and Semmler (2004). Third, since the expected value function is only defined implicitly by the fixed point of the dynamic programming operator, we need to solve a non-linear system of equations in order to obtain an approximation of the expected value. While also under the $EV1$ iid assumption, the expected value is the solution to a fixed point problem, the system becomes much larger in the presence of serial correlation, and thus we discuss suitable methods. Finally, we solve the maximum likelihood problem using a nested fixed point (NFXP) algorithm, which is interconnected with the grid refinement process of the expected value function approximation.

As an application, we estimate the bus engine replacement model of Rust (1987) with serially correlated errors. One motivation for serial correlation in this model is a test for misspecification from the original paper, which leads to the following conclusion:[3]

> "for groups 1, 2, and 3 and the combined groups 1–4 there is strong evidence that (CI) does not hold. The reason for rejection in the latter cases may be due to the presence of 'fixed-effects' heterogeneity which induces serial correlation in the error terms." (Rust, 1987)

Testing for statistical significance of serially correlated errors we find that in some subsamples of the original dataset we can reject serially uncorrelated errors. Also, the parameter estimates vary substantially; their relative sizes however are rather stable. For readability, the development of the algorithm is closely related to the model under consideration; however, note that it is generic with respect to DDCMs with Markov serial dependence in the unobserved state variables.

[3] One can also think of serial correlation as a "generic feature" in this context: In optimal stopping problems, such as the bus engine replacement model, the replacement decision is expected to happen rarely. If the explanatory power of the model in terms of observed states is low, the probability of stopping is small for all possible observed states. Thus, the observed decisions are mostly driven by tail events of the unobserved state variables. However, this fact contradicts the assumption that decisions are modeled to be dynamic, because in a model without serial correlation, these events are unforeseeable, single period shocks. With the introduction of serial correlation, these shocks have persistent effects, which can be anticipated by the agent. For example, a jump in maintenance costs still comes as a surprise to the agent, but — once incurred — its effect on future periods can influence decisions to a large extent.

The remainder of this paper is organized as follows: Section 1.2 describes the bus engine replacement model of Rust (1987), and introduces the notion of serial correlation of the errors, which is used throughout the paper. Section 1.3 first develops a numerical procedure to solve the dynamic programming problem of the agent, then introduces a method, namely the recursive likelihood function integration (RLI) algorithm, to decompose the likelihood function such that it can be computed using highly efficient quadrature rules, and finally describes the likelihood maximization algorithm. Section 1.4 presents the estimation results. Section 1.5 concludes and states the agenda for future research.

1.2 The Bus Engine Replacement Model

In the bus engine replacement model of Rust (1987), an agent repeatedly makes decisions about the maintenance of a fleet of buses: Each period, he observes the state of each of the buses, including mileage, damage, signs of wear, etc. Based on these observations, he decides whether to do regular maintenance work only, or a general overhaul; the latter is usually referred to as a replacement of the engine. While the engine replacement causes a fixed cost of RC plus some random component, the cost of regular maintenance is a function $c(\cdot)$ that is increasing in the current mileage state, plus some random component.

Formally, the agent faces single period costs (or negative utility) for each individual bus

$$u_\theta(i, x_t) + \varepsilon_t(i)\,, \quad u_\theta(i, x_t) = \begin{cases} -RC & \text{if } i = 1 \\ -c(x_t, \theta_1) & \text{if } i = 0 \end{cases} \tag{1.1}$$

where i is the decision variable, with $i = 1$ indicating engine replacement, and $i = 0$ regular maintenance; $\varepsilon_t(i)$ is a random utility component that is observed by the agent for all possible choices before making the actual decision; x_t is the mileage of the individual bus at time t, which is reset to 0 after an engine replacement. The replacement cost RC, as well as the cost function parameter θ_1, are both parameters to be estimated. The maintenance cost function is assumed to be of the form $c(x_t, \theta_1) = 0.001\,\theta_1\,x_t$. From the econometrician's point of view, mileage at the time of decision and the decision itself are observable for each bus and each time period. The random utility component however is only observable to the agent, but not to the econometrician; consequently, it is often referred to as the unobserved state variable.

For the agent, the decision problem is how long to run a bus with regular maintenance only, with increasing costs induced by increasing mileage, and when to replace its engine, thus facing the one-time replacement cost, but at the same time reducing the maintenance costs in the future because mileage is reset to 0. Assuming that the agent behaves dynamically optimally, the Bellman equation defines the value per bus as a function of its mileage state and the random utility components

$$V_\theta(x_t, \varepsilon_t) = \max_{i \in \{0,1\}} \{u_\theta(i, x_t) + \varepsilon_t(i) + \beta \mathbb{E}[V_\theta(x_{t+1}, \varepsilon_{t+1}) | i, x_t, \varepsilon_t]\}. \tag{1.2}$$

The conditional expected continuation value in (1.2) is defined by

$$\mathbb{E}[V_\theta(x_{t+1},\varepsilon_{t+1})|i,x_t,\varepsilon_t] = \int\limits_{(x_{t+1},\varepsilon_{t+1})} V_\theta(x_{t+1},\varepsilon_{t+1})\, Pr(x_{t+1},\varepsilon_{t+1}\,|\,i,x_t,\varepsilon_t,\theta)\,\mathrm{d}(x_{t+1},\varepsilon_{t+1}) \tag{1.3}$$

with subscript θ denoting the dependence of the value function on the parameter values RC and θ_1.

The original model makes the following conditional independence (CI) assumption regarding the joint probability of the state variables:

$$Pr(x_{t+1},\varepsilon_{t+1}|i,x_t,\varepsilon_t,\theta) = Pr(\varepsilon_{t+1}\,|\,x_{t+1})Pr(x_{t+1}\,|\,i,x_t) \tag{1.4}$$

Assumption (1.4) ensures that (i) the mileage state transition is — conditional on the decision i — independent of the random utility component, and (ii) that the random utility components are serially uncorrelated. If the CI assumption holds, and if moreover the random utility components $\varepsilon(i)$ are distributed extreme value type I ($EV1$) iid, the integral in (1.3) has a closed form solution. However, in order to allow for serial correlation in ε, while keeping (i), we assume

$$Pr(x_{t+1},\varepsilon_{t+1}\,|\,i,x_t,\varepsilon_t,\theta) = Pr(\varepsilon_{t+1}\,|\,\varepsilon_t,x_{t+1},\theta)Pr(x_{t+1}\,|\,i,x_t). \tag{1.5}$$

Note that assumption (1.5) allows the transition process of the mileage state, $Pr(x_{t+1}\,|\,i,x_t)$, to be estimated independently from the other model parameters — as in the original model.[4] We use discretized mileage, and thus the integral over future mileage states in (1.3) becomes a sum:

$$\mathbb{E}[V_\theta(x_{t+1},\varepsilon_{t+1})|i,x_t,\varepsilon_t] = \sum_{x_{t+1}}\int\limits_{\varepsilon_{t+1}} V_\theta(x_{t+1},\varepsilon_{t+1})\, Pr(\mathrm{d}\varepsilon_{t+1}\,|\,\varepsilon_t,x_{t+1},\theta)\, Pr(x_{t+1}\,|\,i,x_t) \tag{1.6}$$

A choice for serial correlation in the unobserved state variables that is frequently used in the literature is the $AR(1)$ process. More specifically, similar to Norets (2009), we define

$$\begin{aligned} \varepsilon_t(0) &= \rho\varepsilon_{t-1}(0) + \tilde{\varepsilon}_t(0)\,, \quad \tilde{\varepsilon}_t(0) \sim q(\cdot) \text{ iid} \\ \varepsilon_t(1) &= \qquad\qquad \tilde{\varepsilon}_t(1)\,, \quad \tilde{\varepsilon}_t(1) \sim q(\cdot) \text{ iid} \end{aligned} \tag{1.7}$$

where $q(\cdot)$ is a density function with zero mean, and ρ is the additional parameter of the estimation; furthermore, we assume that $\varepsilon_0(i)$ is distributed with density $q(\cdot)$. Thus, we only assume the random utility component of regular maintenance to be serially correlated.[5] It is

[4] Since one can estimate the mileage transition process $Pr(x_{t+1}\,|\,i,x_t)$ — referred to as parameter θ_3 in the original model — independently from $\theta = \{\theta_1, RC, \rho\}$, and moreover, since it is exactly the same as in Rust (1987) (because it is not affected by the serial correlation in the unobserved state variables) we ignore this aspect of the bus engine replacement model in the remainder of this paper.

[5] While the assumption that serial correlation is only present for regular maintenance utility shocks is computationally more general as we will point out in Section 1.3.1, one can also argue that it is easier to motivate serial correlation as a feature with a real counterpart in this case, because the errors are bus specific by construction; for example, one might think of a bus having some larger damage that persistently increases maintenance costs until the next general overhaul occurs.

important to note that definition (1.7) nests the original model for $\rho = 0$, and the density function $q(\cdot)$ being extreme value type 1 $EV1$.

Given that mileage state x_t and decision i_t are observable for all buses, but random utility components ε_t are not, the aim is to estimate this model's parameter $\theta = \{\theta_1, RC, \rho\}$, given the data $\{x_t, i_t\}_{t=0}^{T}$, by maximum likelihood estimation.

1.3 Computation and Estimation

The following subsections develop the numerical methods necessary to estimate the bus engine replacement model of Rust (1987), with serially correlated unobserved state variables: First, we show how to approximate the solution to the dynamic problem, using numerical quadrature and interpolation over an adaptively refined grid, and solving a large non-linear system of equations. Then, we develop a method to decompose and approximate the integral over the unobserved state variables that appears in the likelihood function, and approximate it using highly efficient Gaussian quadrature, using the recursive likelihood function integration (RLI) algorithm. Finally, we describe a nested fixed point algorithm to obtain maximum likelihood estimates of the model parameters.

Note that this procedure is not specific to the model under consideration, but rather is generic with respect to DDCMs with Markov serial dependence in the unobserved state variables. Also note that the methodology used to approximate the solution to the dynamic problem is independent of the likelihood function integration, for which the new RLI algorithm is proposed in this paper.

1.3.1 The Expected Value Function

From (1.2) it is clear that in order to obtain the value function, we need to compute its conditional expectation. In fact, the computation of the likelihood function actually requires the expected value rather than the value itself (see Section 1.3.2). Thus, this section describes the steps necessary to numerically approximate the expected value as a function of all possible states:

$$EV_\theta(x,\varepsilon) = \sum_{x'} \int_{\varepsilon'} \max_{i\in\{0,1\}} \left\{u_\theta(i,x') + \varepsilon'(i) + \beta EV_\theta(x',\varepsilon')\right\} Pr(\mathrm{d}\varepsilon' \,|\, \varepsilon, \theta)\, Pr(x' \,|\, x) \equiv T(EV_\theta)(x,\varepsilon) \tag{1.8}$$

Keeping the original time structure of the expectation (1.6) in mind, the expectation on the leftmost side of (1.8) is — strictly speaking — taken at time t, while the one on the right hand side (within the max operator) is taken at time $t+1$. But since the value function and its expectation are time invariant, given state (x, ε), the same unknown function EV_θ appears on both the left and the right sides of the equation. Therefore, EV_θ is the solution to the

functional equation

$$EV_\theta(x,\varepsilon) = T(EV_\theta)(x,\varepsilon) \tag{1.9}$$

and thus a fixed point of the non-linear operator T. Moreover, since T can be shown to have the contraction mapping property (Rust, 1988), this fixed point is unique and attractive.

The numerical approximation of (1.8) involves three main computational tasks:[6] First, we need to approximate the integral in (1.8) by numerical quadrature. Second, we have to approximate the continuous function EV_θ by a finite number of parameters, for example by interpolation. Finally, since EV_θ is only defined implicitly as a fixed point of T — and we therefore cannot evaluate it directly — we need to solve for the parameters of the function approximation by solving a non-linear system (or fixed point iteration).

Numerical integration. In contrast to the case of extreme value type I iid distributed unobserved state variables, no closed form solution to the integral (1.8) exists; thus, we have to approximate it by numerical quadrature. A variety of methods for multi-dimensional integration exists; see, for example, chapter 7 of Judd (1998) for an overview, or chapter 4 of Press, Teukolsky, Vetterling, and Flannery (2007) for an implementation oriented approach. Throughout the paper, we use Gaussian quadrature, which is known to be very efficient for the integration of functions that can be well approximated by a polynomial. While this condition is obviously violated for the value function (because of the kink potentially induced by the max-operator), one can still show Gaussian schemes to be convergent for any Riemann integrable function, and, moreover, they are reported to often outperform other widely used integration schemes, even in the presence of singularities; see Judd (1998) and the literature cited therein. Also, Stinebrickner (2000) successfully applied the Gaussian quadrature rules to expected value function approximation for DDCMs with serial correlation.

The n-node Gaussian quadrature rule approximates

$$\int_a^b f(y)w(y)\mathrm{d}y \approx \sum_{i=1}^{n} \omega_i f(y_i) \tag{1.10}$$

where $w(y)$ is a non-negative weighting function with finite integral (including unity for $|a|, |b| < \infty$). The integration nodes y_i are the roots of the degree n polynomial of the family of polynomials that are mutually orthonormal with respect to weighting function $w(y)$.[7] The corresponding weights ω_i are chosen such that every polynomial of degree $2n-1$ is integrated *exactly*; for the corresponding formulas, see, for example, Kythe and Schäferkotter (2005). Since both nodes and weights should be computed to high accuracy, they are often tabulated for some frequently used families of orthonormal polynomials.

[6] Generally, there is one more task necessary, namely maximizing the utility and continuation value, in order to obtain the current value as a function of the states. However, since the choice set is discrete and small, we carry out the maximization by complete enumeration.

[7] A family of polynomials $\{\varphi_k(y)\}_{k=0}^{\infty}$ with inner product $\langle \varphi_k, \varphi_l \rangle = \int_a^b \varphi_k(y)\varphi_l(y)w(y)\mathrm{d}y$ is orthonormal with respect to weighting function $w(y)$ on $[a,b]$ if $\langle \varphi_k, \varphi_l \rangle = 0 \; \forall k,l : k \neq l$, and $\langle \varphi_k, \varphi_k \rangle = 1 \; \forall k$.

When taking expectations of functions of continuous random variables, the integration problem (1.10) arises naturally, with the density function being used as weighting function $w(x)$. Obviously, this approach requires the availability of polynomials that are orthonormal with respect to the density function in use. For some distributions, these families are well known, such as the Hermite polynomials for normally distributed random variables. For most other distributions however, the necessary polynomials (and their roots) are unknown, and have to be computed first. Alternatively, one can map the support of the corresponding density function to $[-1, 1]$ by a change of variable,[8] and approximate the resulting integral using the Gaussian rule based on Legendre polynomials, which are orthonormal with respect to the unity weighting function on $[-1, 1]$. Using this procedure, we found that expectations of extreme value distributed random variables can be approximated quite efficiently.

Directly approximating (1.8) by Gaussian quadrature has a potential caveat, since it would require one to find polynomials that are orthonormal with respect to the conditional probabilities, $Pr(\varepsilon' \,|\, \varepsilon)$, and thus different nodes and weights for each ε. Consequently, we reformulate the integral in (1.8) in terms of the unconditional probability $Pr(\tilde{\varepsilon}'(i))$ (or, equivalently, the density function $q(\cdot)$ of $\tilde{\varepsilon}'(i)$),

$$\begin{aligned}\int_{\tilde{\varepsilon}'(0)} \int_{\tilde{\varepsilon}'(1)} \max\big\{ & u(0, x') + \rho\varepsilon(0) + \tilde{\varepsilon}'(0) + \beta EV_\theta(x', (\rho\varepsilon(0) + \tilde{\varepsilon}'(0), \varepsilon'(1)), \\ & u(1, 1) + \tilde{\varepsilon}'(1) + \beta EV_\theta(1, (0, \varepsilon'(1)))\big\}\, Pr(\mathrm{d}\tilde{\varepsilon}'(1)) Pr(\mathrm{d}\tilde{\varepsilon}'(0))\end{aligned} \tag{1.11}$$

and compute (or look up) one single set of nodes and weights for weighting function $Pr(\tilde{\varepsilon}'(i))$.[9]

Since the integration in (1.11) is of dimension $N = 2$,[10] but Gaussian rules are per se one-dimensional, we use them extended to N dimensions by the product rule, which generalizes (1.10) to N dimensions by

$$\int_{[a,b]^N} f(y^1, \ldots, y^N) \prod_{i=1}^{N} w_i(y^i) \mathrm{d}(y^1, \ldots, y^N) \approx \sum_{i_1=1}^{n} \cdots \sum_{i_N=1}^{n} f(y_{i_1}^1, \ldots, y_{i_N}^N) \prod_{j=1}^{N} \omega_{i_j}^j \tag{1.12}$$

where $f : \mathbb{R}^N \to \mathbb{R}$, $w_i : \mathbb{R} \to \mathbb{R}$ is the weighting function for dimension i, and y_j^i and ω_j^i are the nodes and weights of the corresponding one-dimensional Gaussian rule (indexed by j), applied to dimension i.[11]

[8] For example, if the inverse of the cumulative distribution of a distribution with density $w(y)$, $W^{-1}(y)$ exists, one can apply the following change of variables: $\int_{-\infty}^{+\infty} f(y)w(y) = \int_0^1 f(W^{-1}(y))$.

[9] Equation (1.11) silently assumes that after a replacement, the series of serially correlated unobserved states is reset to its mean, 0. Thus, $\varepsilon(i)$ in the first period after an engine replacement is distributed according to density $q(\cdot)$ again.

[10] The dimension of the integration over the unobserved state variable in DDCMs is usually $(N-1)$-dimensional, because the decisions of the agents in the model are driven by utility differences rather than levels. In this case however, since we assume that serial correlation is only present in one dimension of the error, the reformulation of the model in terms of the differences of errors does not reduce dimensionality. Thus, the integration must be carried out over all the N dimensions.

[11] Note that in order to use the product rule (1.12) to compute expectations, the dimensions of the random variable must be mutually independent. For more general multivariate distributions, see, for example, Jäckel (2005).

Function approximation. Generally, the expected value function is a continuous function of ε, and we need to approximate it as such, but by a finite number of parameters only. Assume for the moment that we can evaluate an unknown function $f(y)$ at arbitrary points. Then, we can choose a set of nodes $y_i \in [a, b]$, and construct an interpolating function $\hat{f}(y)$, such that $f(y_i) = \hat{f}(y_i)\ \forall y_i$. Obviously, we want to choose $\hat{f}(y)$ such that $|f(y) - \hat{f}(y)|$ is "small everywhere", not just at the interpolation nodes y_i. More formally, we want to control the interpolation error $\sup_{y\in[a,b]} |f(y) - \hat{f}(y)|$.

A general, but computationally rather expensive approach to node choice is adaptive procedures: given some interpolant $\hat{f}^{(h)}(y)$, we evaluate the quality of approximation, $|f(y)-\hat{f}^{(h)}(y)|$, at different values of the argument (different from y_i), and we insert new nodes where the approximation quality is poor; then, we construct a new interpolant $\hat{f}^{(h+1)}(y)$ on the set union of old and new nodes. This procedure is iterated until some convergence criterion is met. Adaptive methods are particularly well suited for functions with "difficult" shape properties, for example functions with greatly varying curvature, kinks, or discontinuities, and to explicitly control the approximation error. For the actual interpolation over such a grid, piecewise polynomial interpolation, such as piecewise linear interpolation (PLI) or higher order splines, proved to be a reliable choice.

Since we want to have direct control over the error of the approximation of EV_θ, we choose an adaptive approximation method; in particular, we want to assure uniform approximation quality for different values of θ, in order to compute the corresponding likelihood function values to high accuracy. Therefore, we employ the method of Grüne and Semmler (2004), which repeatedly refines an interpolation grid until a global approximation error criterion is met. At this point, it is important to note that we cannot directly evaluate the true (but unknown) expected value function EV_θ, because it is only implicitly defined by (1.9). Fortunately, to discuss this grid adaption method, it is sufficient to assume that the method is supplied with an approximation $\widehat{EV}_\theta^{(h)}(\,\cdot\,; a)$ from the previous iteration of the adaption process, which is now explicitly parametrized by the finite-dimensional vector $a \in \mathbb{R}^A$. Let $\Gamma_\theta^{(h)}$ be the grid at the beginning of iteration h. For each cell[12] c_l of grid $\Gamma_\theta^{(h)}$, we approximate the solution to the following optimization problem:[13]

$$\eta_l = \max_{\varepsilon\in c_l} |\widehat{EV}_\theta^{(h)}(x, \varepsilon; a) - T(\widehat{EV}_\theta^{(h)})(x, \varepsilon; a)| \tag{1.13}$$

Then, Grüne (1997) showed that the maximum error over all cells, $\eta = \max_l\{\eta_l\}$, defines an approximation error bound by

$$\max_{x\in X,\varepsilon\in R^N} |EV_\theta(x, \varepsilon) - \widehat{EV}_\theta^{(h)}(x, \varepsilon; a)| \le \eta\frac{1}{1-\beta} \tag{1.14}$$

[12] In this context, cell c_i of an n-dimensional grid Γ is defined as the hypercube spanned by $\{y_j \in \Gamma : y_i^k \le y_j^k \le \min_l\{y_l^k : y_i^k < y_l^k\}, k = 1, \ldots, n\}$, where y^k is the kth element (dimension) of the vector y.

[13] Note that since the model is already discretized in terms of mileage state x, finding the maximum error within each cell does not explicitly involve x; rather, one has to carry out the error estimation for all possible mileage states independently.

where EV_θ represents the true (but unknown) expected value function. The method of Grüne and Semmler (2004) inserts new nodes into those cells c_l where the corresponding error η_l is larger than some threshold. Finally, we construct new interpolant $\widehat{EV}_\theta^{(h+1)}(\,\cdot\,;a)$ on the refined grid $\Gamma_\theta^{(h+1)}$. (In order to parametrize it, we need to solve for the fixed point (1.9), which we will discuss shortly.) This procedure is repeated until the maximum (global) approximation error $\eta(1-\beta)^{-1}$ is smaller than the desired approximation error, $\bar{\eta}$.

One particular advantage of the method of Grüne and Semmler (2004) is that it not only allows for refinement, but easily extends to grid coarsening, by identifying and removing nodes that do not increase approximation accuracy. Combining coarsening and refinement, we can construct a grid *updating* procedure, which can be integrated with a nested fixed point algorithm (NFXP). In NFXP, the likelihood maximization ("outer loop") repeatedly feeds different values of θ into the expected value function approximation ("inner loop"); thus, rather than building up from scratch an interpolant for each new value of $\theta^{(k+1)}$, it can be obtained from updating an interpolant that has previously been built for some other value $\theta^{(k)}$ (see Section 1.3.3 below).

Note that due to the fact that serial correlation is only allowed in $\varepsilon(0)$, $EV_\theta(x,\varepsilon)$ is constant in $\varepsilon(1)$. Consequently, we only need to approximate it as a one-dimensional function of $\varepsilon(0)$. Therefore, we can use piecewise linear interpolation to construct $\widehat{EV}_\theta$. However, the methodology generalizes to higher dimensions by replacing PLI with multi-dimensional interpolation.

Finally note that, since — in this formulation of the model — mileage has been discretized, we need to approximate EV_θ as a separate continuous function of ε for each mileage state $x \in X$ simultaneously; thus, $\widehat{EV}_\theta(\,\cdot\,;a)$ is really a set of interpolants. If, in contrast, mileage would enter the model as a continuous variable, $\widehat{EV}_\theta(\,\cdot\,;a)$ would rather be a single 2-dimensional interpolant. However, discrete mileage is necessary to nest the original model without serial correlation as a special case.

Non-linear system. The last few paragraphs discussed the choice of a function approximation scheme and interpolation grid creation, but left out how to actually evaluate the unknown function EV_θ, which is only implicitly defined as the fixed point of T. While this fixed point is generally a continuous function, its substitution by an approximating interpolant $\widehat{EV}_\theta(\,\cdot\,;a)$ simplifies the problem to a non-linear system of D equations in A unknowns,

$$\widehat{EV}_\theta(x,\varepsilon;a) = T(\widehat{EV}_\theta)(x,\varepsilon;a) \quad \forall (x,\varepsilon) \in \Gamma_\theta, a \in \mathbb{R}^A \tag{1.15}$$

where D is the number of elements in Γ_θ, and thus each $(x,\varepsilon) \in \Gamma_\theta$ defines one equation of (1.15), and the parameters a of the interpolant are the variables. From the parameter vector a^* that solves (1.15), we can directly construct the interpolant $\widehat{EV}_\theta(\,\cdot\,;a^*)$. This procedure is known as collocation, which is a particular variant of a projection method for the approximation of functions that are defined by functional equations; see Judd (1998), chapter 11. Finally, we compute the approximation error of $\widehat{EV}_\theta(\,\cdot\,;a^*)$ as defined by (1.14); if it is sufficiently small (smaller than $\bar{\eta}$), we accept our approximation of EV_θ; otherwise, we refine the interpolation grid Γ_θ, and solve (1.15) for the new grid.

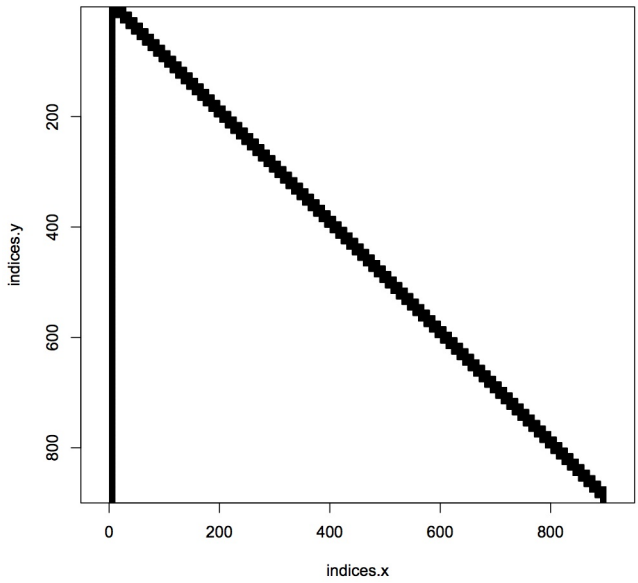


Figure 1.1: Sparseness pattern of the Jacobian of the non-linear system (1.16).

Similar to Rust (1987), we use methods that directly solve the non-linear system

$$\widehat{EV}_\theta(x,\varepsilon;a) - T(\widehat{EV}_\theta)(x,\varepsilon;a) = 0 \quad \forall (x,\varepsilon) \in \Gamma_\theta, a \in \mathbb{R}^A \tag{1.16}$$

to high accuracy. Given the accuracy needs of our application, Newton (or quasi-Newton) methods are particularly interesting, because they show quadratic (superlinear) convergence close to the solution under some conditions.[14] However, these methods require the evaluation of the Jacobian matrix J of the non-linear system (1.16), which is generally of size D^2, and thus can be prohibitively expensive to compute for large systems. In particular, given an adaptively refined grid, the size of J can become an issue since the number of equations of (1.16) is defined by the number of nodes in Γ_θ, and thus the system grows larger as the grid is refined. However, analogously to the original model, if the Markov transition matrix of the discrete states is sparse, J is also sparse; thus, using (quasi-)Newton methods can still be feasible because the number of non-zero elements in the Jacobian grows much more slowly than the number of grid nodes. Figure 1.1 illustrates the sparseness pattern of our problem.

To numerically solve the fixed point problem (1.9), we either use the "ipopt" package (Wächter and Biegler, 2005), in conjuction with the "pardiso" sparse linear solver (Schenk and Gärtner, 2004), or the quasi-Newton trust-region method of the R-package "nleqslv" (Hasselman, 2014), depending on the size of the problem.

Figure 1.2 plots an example of the expected value function, where each of the black lines

[14] Loosely speaking, quadratic convergence means that, close to the solution, the number of correct digits of the result roughly doubles in every Newton step. More formally, suppose that for $f : \mathbb{R}^n \to \mathbb{R}^n$, a solution y^* to the system $f(y^*) = 0$ exists, the Jacobian function $J : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is Lipschitz continuous, and the Jacobian matrix at the solution, $J_f(y^*)$ is non-singular. Then, if $y^{(0)}$ is sufficiently close to the solution y^*, the residual decays quadratically for each Newton iteration, thus $\exists K > 0 : \|y^{(k+1)} - y^*\| \leq K\|y^{(k)} - y^*\|^2$.

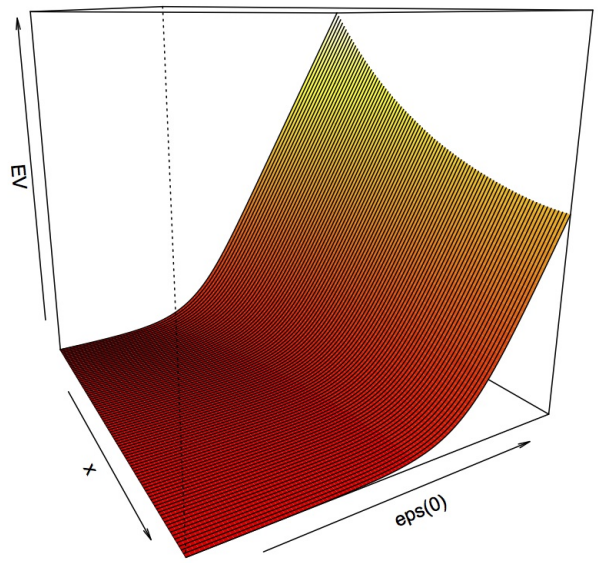


Figure 1.2: The expected value function $EV_\theta(x, \varepsilon)$ for $\rho = 0.6$, $RC = 14$, $\theta_1 = 2$, and the density of $\tilde{\varepsilon}(i)$, $q(\cdot)$, being $EV1$.

represents the expected value as a function of $\varepsilon(0)$, for a particular value x. We want to emphasize again that the procedure to compute an approximation of $EV_\theta(x, \varepsilon)$ as presented in this section easily generalizes to other models, with an arbitrary number of decisions N, and serial correlation in all dimensions of the unobserved state variables, by choosing a multi-dimensional interpolation scheme.

1.3.2 The Likelihood Function

In this section, we derive the likelihood function for the bus engine replacement model with serially correlated unobserved state variables, and formulate it such that the dimensionality of the numerical integration only depends on the number of choices N, and not on the time horizon of the observation, T. In a second step, we provide a numerical procedure to solve this formulation by recursive likelihood function integration (RLI) to high accuracy, using standard deterministic quadrature rules.[15] It is important to note that this reformulation is not specific to the Rust (1987) model, but generically applies to DDCMs with Markov serial dependence in the unobserved state variables.

The likelihood function of one individual bus derives as follows:

$$L(\,\theta\,|\,\{x_t, i_t\}_{t=0}^{T}) = \int_{\varepsilon_0,\ldots,\varepsilon_T} \cdots \int Pr(\{x_t, i_t, \varepsilon_t\}_{t=0}^{T}\,|\,\theta)\;\mathrm{d}\varepsilon_0 \ldots \mathrm{d}\varepsilon_T \tag{1.17}$$

[15] This is not to be confused with the recursive maximum likelihood estimation (RMLE) algorithm of Kay (1983) for the estimation of AR processes, which allows one to recursively update maximum likelihood estimates to higher order AR models.

The likelihood function of the full panel computes as the product of the likelihood functions of the individual buses, since the state variables are assumed to be independently distributed across buses. Incorporating the assumption that all state transitions are Markov, we can factorize the probability of observing a particular time series as

$$Pr(\{x_t, i_t, \varepsilon_t\}_{t=0}^{T} \,|\, \theta) = \prod_{t=1}^{T} Pr(x_t, i_t, \varepsilon_t \,|\, x_{t-1}, i_{t-1}, \varepsilon_{t-1}, \theta). \tag{1.18}$$

We can further decompose the joint transition probability in (1.18), using the fact that, given x_t and ε_t, i_t is independent of i_{t-1}, ε_{t-1}, and x_{t-1}, as well as incorporating assumption (1.5):

$$Pr(x_t, i_t, \varepsilon_t \,|\, x_{t-1}, i_{t-1}, \varepsilon_{t-1}, \theta) = Pr(i_t \,|\, x_t, \varepsilon_t, \theta) Pr(\varepsilon_t \,|\, i_{t-1}, \varepsilon_{t-1}, \theta) Pr(x_t \,|\, x_{t-1}, i_{t-1}) \tag{1.19}$$

For notational simplicity, we define

$$m_{it} \equiv u_\theta(i, x_t) + \beta \mathbb{E}[V_\theta(x_{t+1}, \varepsilon_{t+1}) | i, x_t, \varepsilon_t]. \tag{1.20}$$

While $Pr(\varepsilon_t \,|\, i_{t-1}, \varepsilon_{t-1}, \theta)$ is determined by (1.7) and $Pr(x_t \,|\, x_{t-1}, i_{t-1})$ is estimated independently (and therefore omitted from now on),[16] the conditional decision probability $Pr(i_t \,|\, x_t, \varepsilon_t, \theta)$ is given by

$$Pr(i_t = 1 \,|\, x_t, \varepsilon_t(0), \varepsilon_t(1), \theta) = \mathbb{1}(m_{1t} + \varepsilon_t(1) > m_{0t} + \varepsilon_t(0)) \tag{1.21}$$

where $\mathbb{1}(\cdot)$ is the index function that is equal to one if its argument is true, and zero otherwise; note that the conditional decision probabilities are actually degenerate, because — loosely speaking — there is no randomness left, given ε_t.

Finally, exploiting the Markov structure for the integration, and dropping parameter dependence for better readability, we can write the likelihood function (1.17) as

$$\int_{\varepsilon_0} \cdots \int_{\varepsilon_{T-1}} \prod_{t=1,\ldots,T-1} Pr(i_t \,|\, x_t, \varepsilon_t) Pr(\varepsilon_t \,|\, i_{t-1}, \varepsilon_{t-1}) \int_{\varepsilon_T} Pr(i_T \,|\, x_T, \varepsilon_T) Pr(\varepsilon_T \,|\, i_{T-1}, \varepsilon_{T-1}) \, \mathrm{d}\varepsilon_0 \ldots \mathrm{d}\varepsilon_{T-1} \mathrm{d}\varepsilon_T \tag{1.22}$$

To numerically approximate (1.22), we define the function

$$g_t(\varepsilon) = \begin{cases} 1 & t > T \\ \displaystyle\int_{\varepsilon'} Pr(i_t \,|\, x_t, \varepsilon') Pr(\varepsilon' \,|\, i_{t-1}, \varepsilon) g_{t+1}(\varepsilon') \, \mathrm{d}\varepsilon' & \text{otherwise} \end{cases} \tag{1.23}$$

Now, given $g_{t+1}(\varepsilon)$, we can numerically approximate the function $g_t(\varepsilon)$ using both numerical integration and function approximation. Since $g_t(\varepsilon)$ is known to be unity for $t > T$, we can use backward iteration starting from $g_T(\varepsilon)$ to solve for $g_0(\varepsilon)$, which is the approximation of

[16] Since one can estimate the mileage transition probabilities separately, they only add a multiplicative constant to the likelihood function of $\theta = \{\theta_1, RC, \rho\}$. Thus, we omit the corresponding term of the likelihood function (and one should do so in the actual maximization for scaling reasons).

the likelihood function $L(\theta \mid \cdot)$.[17] Note that this procedure is analogous to solving for the value function of a finite horizon, discrete time dynamic programming problem by backward iteration. Algorithm 1.1 gives a formal description of the procedure.[18]

Algorithm 1.1 Computation of the likelihood function (1.22) by recursive likelihood function integration (RLI).

1: $\Gamma \leftarrow$ initialize grid over support of ε with D elements
2: $\hat{g}(\cdot) \leftarrow$ initialize interpolant with nodes $\{(e, \tilde{g}_e)\}_{e \in \Gamma}$ to unity
3: **for** $t = T, \ldots, 1$ **do**
4: **for** $e \in \Gamma$ **do**
5: $\tilde{g}_e \leftarrow$ approximate $\int_{\varepsilon'} Pr(i_t \mid x_t, \varepsilon') Pr(\varepsilon' \mid i_{t-1}, e) \hat{g}(\varepsilon') \, \mathrm{d}\varepsilon'$
6: **end for**
7: $\hat{g}(\cdot) \leftarrow$ construct interpolant with nodes $\{(e, \tilde{g}_e)\}_{e \in \Gamma}$
8: **end for**

Note that each integral over ε_t is generally still N-dimensional. Thus, the procedure decomposes the $T \cdot N$-dimensional integral of (1.17) to an N-dimensional integration that is repeated $D \cdot T$ times, where D is the number of nodes used for function approximation of g_t. Since the computational complexity of deterministic numerical integration is generally exponential in the number of dimensions, this reduction is highly desirable even for large D, because it enters the complexity of the overall algorithm linearly[19]

$$O(\exp(T \cdot N)) \gg O(D \cdot T \exp(N)) \tag{1.24}$$

Given that serial correlation is only allowed in some dimensions, but not all, we can potentially replace parts of the integral in (1.23) by a closed form solution; this is particularly the case if the cumulative distribution of those unobserved state variables that are not serially correlated does have a closed form. Recall that the integration over ε_t is really N-dimensional, thus 2-dimensional in the model under consideration:

$$\int_{\varepsilon_t(0)} \int_{\varepsilon_t(1)} Pr(\varepsilon_t(0) \mid i_{t-1}, \varepsilon_{t-1}(0)) \; Pr(\varepsilon_t(1)) \; Pr(i_t \mid x_t, \varepsilon_t(0), \varepsilon_t(1)) \, \mathrm{d}\varepsilon_t(1) \mathrm{d}\varepsilon_t(0) \tag{1.25}$$

Using (1.21), we can write the integral over $\varepsilon_t(1)$ in terms of its cumulative distribution function

[17] Recursive computation of the likelihood function for serially correlated unobserved Markov states is not a new idea in general. However, to the best of our knowledge, its application has been limited to discrete state spaces, and therefore with no need for numerical quadrature or function approximation; see, for example, Cosslett and Lee (1985) for the estimation of models with Markov regime switching.

[18] Algorithm 1.1 is generic with respect to both the numerical integration scheme and the function approximation schemes, as long as the latter depend on function evaluations only. Also, it can be applied analogously to the case of discrete (or discretized) error processes.

[19] In this context, the $O(f(y))$ notation for the computational complexity of an algorithm reads as follows: There exists a constant $K > 0$ such that the number of iterations needed for an algorithm to complete a task of size y is bounded by $K \cdot f(y)$.

F,

$$\begin{aligned}&\int_{-\infty}^{\infty} \mathbb{1}(\varepsilon_t(1) > m_{0t} - m_{1t} + \varepsilon_t(0))\ Pr(\varepsilon_t(1))\ \mathrm{d}\varepsilon_t(1) \\ &= \int_{m_{0t}-m_{1t}+\varepsilon_t(0)}^{\infty} Pr(\varepsilon_t(1))\ \mathrm{d}\varepsilon_t(1) = 1 - F(m_{0t} - m_{1t} + \varepsilon_t(0))\end{aligned} \tag{1.26}$$

which no longer involves numerical quadrature if an analytical formula for F exists.

For the actual computations we use Gaussian quadrature as outlined in the previous section (in the context of expected value function approximation). Note that while we write all integrals in this section as integrals over ε for simplicity, we have to reformulate them in terms of $\tilde{\varepsilon}$ by a linear change of variables in order to approximate them by Gaussian quadrature (see Section 1.3.1). Also, for numerical reasons, we chose a slightly different change of variables to map the integration domain from $[-\infty, \infty]$ to $[-1, 1]$, (see Judd, 1998, p. 204). Furthermore, we use Akima splines (Akima, 1970) to approximate the integral over ε_t as a function of ε_{t-1}.

1.3.3 Likelihood Function Maximization

Obtaining the maximum likelihood estimate of θ, given data $\{x_t, i_t\}_{t=0}^{T}$, requires us to find a solution to the following two problems *simultaneously*:

$$\hat{\theta} = \arg\max_{\theta}\ L(\,\theta \,|\, \{x_t, i_t\}_{t=0}^{T}, \widehat{EV}_\theta) \tag{1.27}$$

$$\widehat{EV}_\theta(x, \varepsilon; a) = T(\widehat{EV}_\theta)(x, \varepsilon; a) \quad \forall (x, \varepsilon) \in \Gamma_\theta, a \in \mathbb{R}^A \tag{1.28}$$

While there exist methods that directly solve (1.27) and (1.28) simultaneously as a constrained optimization problem, namely the mathematical programming with equilibrium constraints (MPEC) approach to DDCM estimation of Su and Judd (2012), we use the well known nested fixed point (NFXP) approach of Rust (1988).[20] In NFXP, the likelihood maximization is performed as a repeated two step procedure: First, given a parameter guess $\theta^{(k)}$, one computes the expected value function $EV_{\theta^{(k)}}$ as a fixed point of operator T by solving (1.28). Second, one evaluates the likelihood function for $\theta^{(k)}$, using the approximation of $EV_{\theta^{(k)}}$ just previously obtained. The optimization algorithm then constructs a new parameter guess $\theta^{(k+1)}$, and the

[20]The MPEC approach to DDCM estimation of Su and Judd (2012) "combines" the solution of the fixed point and the maximization of the likelihood by solving the original constraint formulation of the likelihood maximization problem (1.27). This procedure is considered to be more efficient in some cases, because it does not require one to solve the fixed point equation (1.9) for each parameter guess, even if it is far away from the solution; rather, it imposes the fixed point condition to hold only at the solution. However, directly integrating MPEC with adaptive interpolation grids creates two potential problems: First, adding a grid node corresponds to adding a constraint to the optimization problem, while the optimization algorithm runs. Second, adaptive methods usually require the approximation of an iteration to be completed in order to compute the approximation quality for the insertion decision, which in our case is not possible until (1.9) has been solved, which in turn contradicts the MPEC idea.

procedure starts again by approximating $EV_{\theta^{(k+1)}}$. This is iterated until convergence of the maximization algorithm.[21] Thus, (1.27) can be solved as an unconstrained problem.

Recall that the interpolation grid $\Gamma_{\theta^{(k)}}$, over which the corresponding approximating interpolant $\widehat{EV}_{\theta^{(k)}}(\,\cdot\,;a)$ satisfies some error bound $\bar{\eta}$, depends on $\theta^{(k)}$. Thus, each step of the maximization routine, from $\theta^{(k)}$ to $\theta^{(k+1)}$, requires one to iteratively update the grid from $\Gamma_{\theta^{(k)}}$ to $\Gamma_{\theta^{(k+1)}}$, until the maximum approximation error of $\widehat{EV}_{\theta^{(k)}}(\,\cdot\,;a)$ is bounded by $\bar{\eta}$ again; this procedure ensures that for each likelihood function evaluation, the approximation error of the corresponding expected value function is controlled.[22]

Algorithm 1.2 summarizes the nested fixed point algorithm to solve (1.27).

Algorithm 1.2 Nested fixed point algorithm with adaptive grid updating.

initialize θ, Γ_θ, a
while not converged **do**
    while $\eta(1-\beta)^{-1} > \bar{\eta}$ **do**
        solve $\widehat{EV}_\theta(x,\varepsilon;a) = T(\widehat{EV}_\theta)(x,\varepsilon;a) \quad \forall (x,\varepsilon) \in \Gamma_\theta, a \in \mathbb{R}^A$
        update Γ_θ (coarsening and refinement)
    end while
    evaluate $L(\theta)$
    compute next θ
end while

For the model under consideration, the maximization of the likelihood function is a non-linear, partially box-constrained optimization problem with three free parameters. To numerically solve this problem, we employ the model-based, derivative-free trust-region method "bobyqa" (Powell, 2009).[23]

1.4 Estimation Results

The original dataset of Rust (1987) consists of monthly odometer readings and engine replacement decisions for a fleet of 162 buses, subdivided into 8 groups depending on their manufacturer and model. Since buses are heterogeneous across groups, it is common to create different subsamples to estimate the parameters of model (1.1); we follow the literature by estimating three subsamples separately, consisting of groups $\{1,2,3\}$, $\{1,2,3,4\}$, and $\{4\}$.

[21] Since the fixed point of T is usually obtained using an iterative method, solving the dynamic problem is often referred to as the "inner loop" in this context, while the maximization procedure is referred to as the "outer loop".

[22] Controlling the maximum approximation error does not imply that it is constant over the maximization procedure. Rather, we choose $\bar{\eta}^{(k)}$ to be decreasing in the iterations of the optimizer, in order to compute the fixed point to lower accuracy far away from the solution, but to high accuracy close to it.

[23] According to Powell (2009), the name "bobyqa" is an acronym for "Bound Optimization BY Quadratic Approximation."

Table 1.1 shows the size of the panel for each group under consideration.

Bus group	Number of buses (M)	Observation horizon (months)	Total number of observations	Number of replacements
1	15	25	360	0
2	4	49	192	0
3	48	70	3,312	27
4	37	117	4,292	33
Total	104		8,156	60

Table 1.1: Number of buses, observation time horizon in months, total number of observations, and number of observed engine replacements for each bus group.

As in Rust (1987), we discretize mileage in "bins" of 5,000 miles each.[24] The highest possible mileage state is 90 (which corresponds to 450,000 miles),[25] formally $x \in X = \{1, \ldots, 90\}$. We assume the mileage transition to follow a Markov process (conditional on the replacement decision), for which we estimate the parameters independently. We parametrize the discount factor by $\beta = 0.9999$ as in the original paper.

Before presenting the results of the estimation, we verify the estimation procedure presented in Section 1.3: First, Table 1.2 presents a partial reproduction of Table IX of Rust (1987), without serial correlation, but still numerically integrating both the expected value and the likelihood function. We conclude that, for the case without serial correlation, we are well able to replicate the original estimates.

Second, we carry out an extensive Monte Carlo study, where we simulate the model from Section 1.2 to create many data sets of different sizes (number of buses),[26] for both densities, extreme value type 1 $EV1$ and standard normal $N(0, 1)$, and estimate the parameters from these data sets using NFXP together with the RLI algorithm. The objective is to investigate the ability of the method to recover the parameters from the data, for which we know the true values in the case of simulated data.[27] Therefore, for each data set size $M \in \{100\,,\, 1000\,,\, 10000\}$, and for both densities, we create 200 datasets; on each data set, we run an estimation with and without allowing for serial correlation (i.e. setting $\rho = 0$). Table 1.3 presents the results of this

[24] By discretizing into bins of 5,000 miles we mean that the original mileage $\tilde{x}$ transforms into a mileage state $x = \lceil \tilde{x}/5,000 \rceil$, with the ceiling function $\lceil \tilde{y} \rceil = \min\{y \in \mathbb{N} : y \geq \tilde{y}\}$.

[25] If a bus ever reaches the maximum mileage state, we assume it to stay there until engine replacement. Although no bus in any of our subsamples ever reaches the maximum mileage state, it still has relevance for the solution of the dynamic problem of the agent, who takes this possibility into account when solving his infinite horizon dynamic optimization problem.

[26] Note that we refer to data set size as the number of buses in a dataset, or, equivalently, as the number of replacement observations, as we simulate each bus until replacement.

[27] The values for the parameters are chosen such that they resemble the estimates for the largest subset of the original dataset for the respective distribution, as reported below.

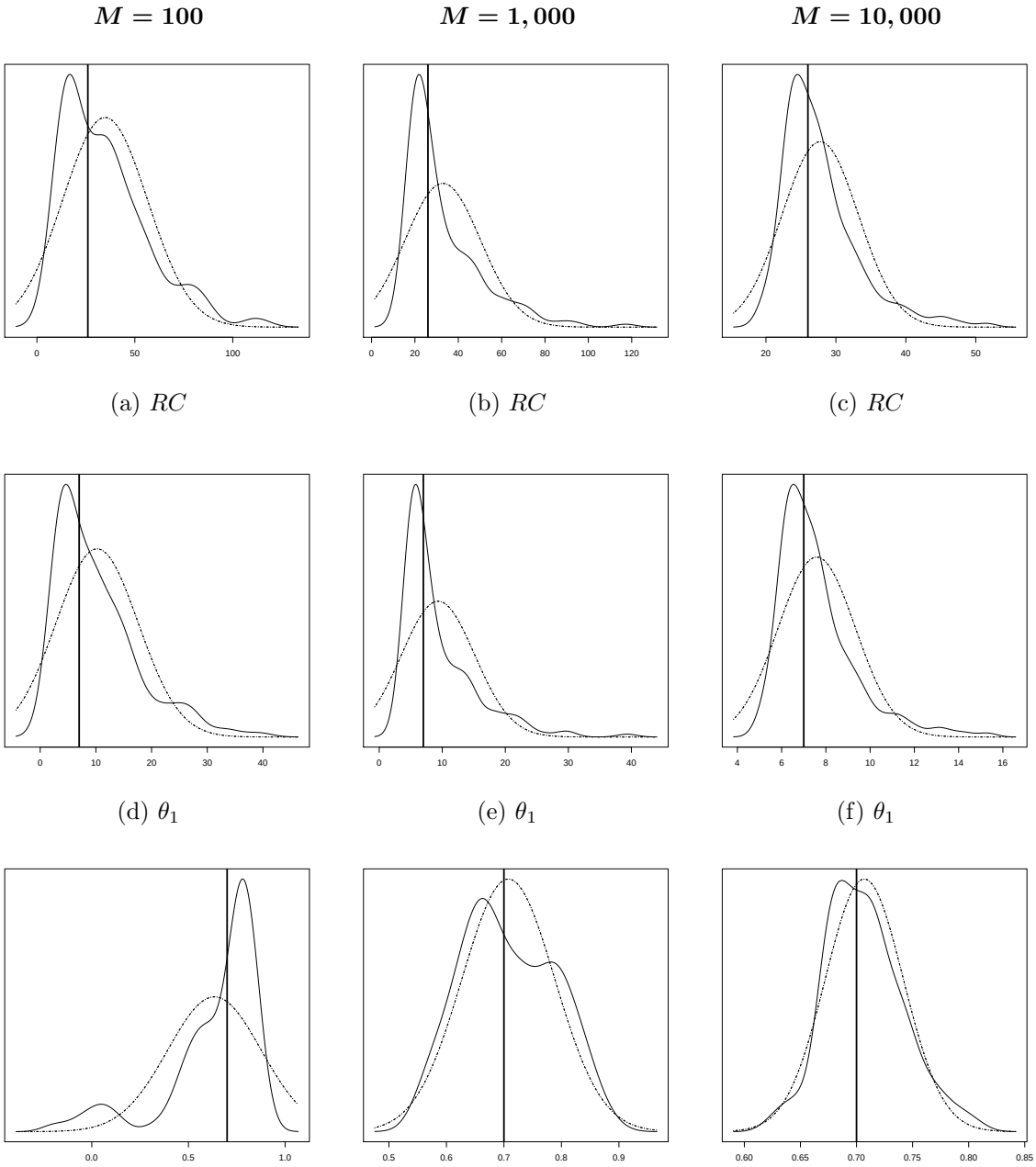
	Bus groups 1–3		Bus groups 1–4		Bus group 4	
	Rust (1987)	Estimated	Rust (1987)	Estimated	Rust (1987)	Estimated
RC	11.7270	11.7266	9.7558	9.7560	10.0750	10.0749
	(2.602)	(1.928)	(1.227)	(0.898)	(1.582)	(1.351)
θ_1	4.8259	4.8257	2.6275	2.6276	2.2930	2.2929
	(1.792)	(1.366)	(0.618)	(0.469)	(0.639)	(0.554)
ρ	–	–	–	–	–	–
L	-2,708.366	-2,708.366	-6,055.250	-6,055.250	-3,304.155	-3,304.156

Table 1.2: Replication of Table IX of Rust (1987) for all subsamples reported therein; L is the value of the log-likelihood function at the solution; $\beta = .9999$.

Monte Carlo study by reporting means and standard deviations of the respective estimates. We also report mean and standard deviation of the likelihood ratio test with the null hypothesis of absence of serial correlation, carried out on the individual data set level. Figures 1.3 and 1.4 finally plot a kernel smoothing estimation of the distribution of the estimates, together with the true parameter values, and the density of the normal distribution with mean and standard deviation as reported in Table 1.3.

From this Monte Carlo study we draw the following conclusions: First, while the method seems to slightly overestimate both cost parameters, the true parameters are always well within one standard deviation. Also, in case of $EV1$ distributed $\tilde{\varepsilon}$ where the overestimation is most apparent, the mean of the estimates clearly gets closer to the true values as we increase the data set size. For the serial correlation parameter ρ, we observe almost perfect recovering of the true parameter value for large data sets, in the $EV1$ case even for moderate data set sizes. Comparing the estimates with serial correlation to the case where serial correlation is ruled out by setting $\rho = 0$, we see that the parameter estimates vary considerably. However, looking at the (probably more relevant) ratio of the cost parameters, we find that the misspecification bias is small in the $EV1$ case, but moderate in the $N(0, 1)$ case. Testing for the statistical significance of the increase in quality of fit by allowing for serially correlated errors using the likelihood ratio test, we find that given a data set of 100 buses (which is comparable to the largest subset of the original data presented above), it is often impossible to reject the no-serial-correlation hypothesis at a reasonable significance level, even if the true model features serial correlation as in (1.7). While in the $EV1$ case, significance increases vastly for the larger data sets under consideration, the model with normal $\tilde{\varepsilon}$ has a surprisingly low increase in quality of fit, along with relatively large p-values even for large data sets.

Turning our attention to Figures 1.3 and 1.4, we notice that for the smaller data sets the distribution of the parameter estimates is clearly not normal. Moreover, it even appears to be bimodal, with one solution being the no serial correlation case. However, since for the large

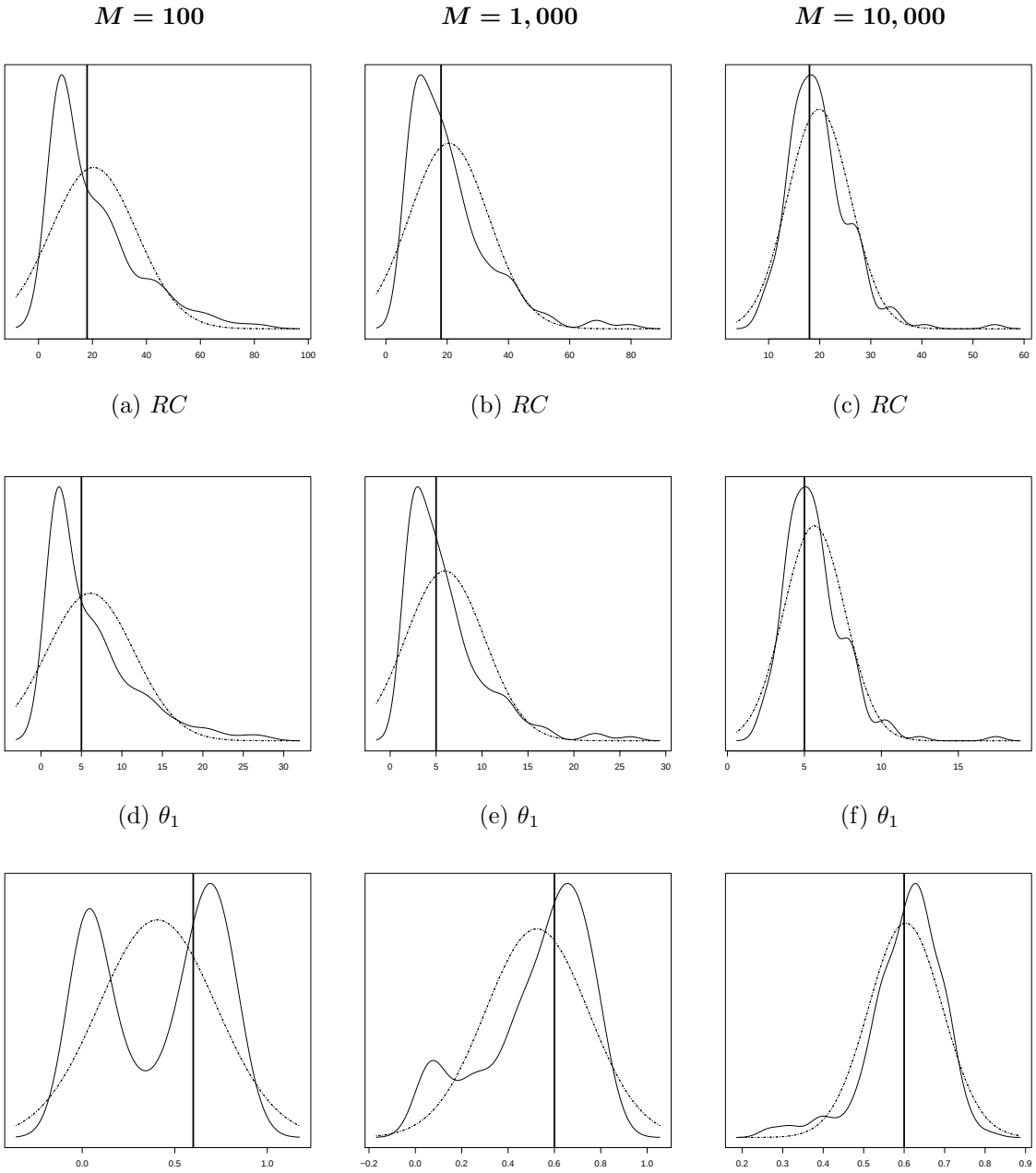


(a) RC (b) RC (c) RC

(d) θ_1 (e) θ_1 (f) θ_1

(g) ρ (h) ρ (i) ρ

Figure 1.3: Distributions of the maximum likelihood estimates from 200 artificial data sets of different sizes, with density $q(\cdot)$ being extreme value type 1 $EV1$. The bold solid vertical lines denote the true parameter value; the thin solid lines are kernel smoothing estimates of the distributions of the parameter estimates; the dash-dotted lines depict normal distributions with mean and standard deviation of the respective estimates. The lefthand column uses data sets of 100 buses each, the center column 1,000 buses, and the righthand column 10,000 buses.



(a) *RC* (b) *RC* (c) *RC*

(d) θ_1 (e) θ_1 (f) θ_1

(g) ρ (h) ρ (i) ρ

Figure 1.4: Distributions of the maximum likelihood estimates from 200 artificial data sets of different sizes, with density $q(\cdot)$ being standard normal $N(0, 1)$. The bold solid vertical lines denote the true parameter value; the thin solid lines are kernel smoothing estimates of the distributions of the parameter estimates; the dash-dotted lines depict normal distributions with mean and standard deviation of the respective estimates. The lefthand column uses data sets of 100 buses each, the center column 1,000 buses, and the righthand column 10,000 buses.

data sets, the distributions apparently become closer to the density of the normal distribution, especially for the cost parameters in the $N(0,1)$ case, and for the serial correlation parameter in the $EV1$ case, it appears that the estimators might actually be asymptotically normally distributed.

At this point it is worthwhile commenting on the sources and potential impact of numerical truncation error: First, we found that the likelihood function is very flat on the right tail of the cost parameter distribution; keeping in mind that the stopping criterion of an optimization algorithm introduces a truncation error of its own, this could well explain the local modes on these tails. Second, while we use a specific Gaussian rule for the normally distributed $\tilde{\varepsilon}$, namely the Gauss–Hermite rule, we use a change of variable to adapt the Gauss–Legendre rule with uniform weighting to the extreme value distribution, which most likely does not perfectly cover the fat tail of this distribution; also, both schemes are applied to a function that is not globally continuously differentiable because of the max-operator. Consequently, in flat regions of the likelihood function, the approximation error of the EV function from both the integration and the interpolation error might dominate the (true) change in the objective function value, making it hard for the solver to distinguish between real progress and computational noise. Thus, it is difficult to judge which effects come from the model structure, and which are numerical artifacts. Consequently, the sources and the impact of numerical error — including the error from the likelihood function approximation — will be subject to further research.

Table 1.4 finally presents the estimation results using the original dataset of Rust (1987), again for both $EV1$ and normally distributed $\tilde{\varepsilon}$. As for the artificial data, we observe that in the $EV1$ case, while the parameter estimates in the presence of serial correlation are substantially different from the estimates without serial correlation, the ratio of engine replacement cost to the regular maintenance cost parameter is relatively stable; thus, the trade-off for the decision maker has not changed much quantitatively. Performing a likelihood ratio test to compute the statistical significance of the quantitative changes induced by the introduction of serial correlation, we find that only on the largest subsample of the dataset (bus groups 1–4) can we reject the hypothesis of no serial correlation at a reasonable significance level. Also, comparing the parameter estimates from the unrestricted model to their counterparts from the restricted model individually, we observe that the difference for the cost parameters is roughly within the standard error;[28] only the serial correlation parameter itself deviates by more than two standard errors. The case of normally distributed $\tilde{\varepsilon}(i)$ yields similar results, with two notable differences: First, not only do the cost parameter values change substantially, but also their ratios and thus the trade-off for the decision maker. This is because the variance of the error is no longer normalized, but rather depends positively on ρ, which also tends to cause identification issues. Second, while all parameter estimates from the largest subsample (bus groups 1–4) in the unrestricted case deviate by more than one standard error from their

[28] For the estimation of the standard errors from the original data set, we use the inverse of the negative Hessian of the likelihood function at its maximum, $(-H(\hat{\theta} \mid \{x_t, i_t\}_{t=0}^{T}))^{-1}$, which is approximated using finite differences.

restricted counterparts, the estimates from the smaller subsamples are within the standard error even for the serial correlation parameter. Carrying out a likelihood ratio test, we cannot reject the hypothesis of no serial correlation at a reasonable significance level for any of the subsamples in the normal case.

We interpret the change of the ratio of the cost parameters in this particular model as follows (as an example, we assume the ratio in the restricted model to be larger than in the unrestricted one): If we ignore serial correlation, the relative costs of regular maintenance are underestimated. Consequently, using the true relative costs in a model without serial correlation, we would predict more (or, equivalently, earlier) engine replacement than we find in the data. Thus, allowing for serial correlation explains why we do not observe more frequent engine replacement, given the high (true) relative costs of regular maintenance. Conversely, in a model with serial correlation, but based on the biased relative costs estimates, we would predict the buses to run for too long without engine replacement.

Assessing the question of the statistical significance of the estimates from the original data set is difficult though. First, from our experiments with artificial data sets we learned that the results are rarely significant for small samples, even if the true model features serial correlation as defined by (1.7). Consequently, given the number of buses in the original data set, significance as for groups 1–4 with extreme value distributed $\tilde{\varepsilon}(i)$ is not what we can generally expect. Second, we still cannot conclude that the serial correlation we found in the data is really coming from an unobserved source, as different bus groups are pooled together for two of the three subsamples, thus creating a heterogeneous sample that is treated as homogeneous by the model. Consequently, as long as we do not find the serial correlation *within* one single bus group to be significant, these estimations have to be taken with a grain of salt.

1.5 Conclusion

This paper developed a method to efficiently estimate dynamic discrete choice models in the presence of serial correlation in the unobserved state variables. First, to approximate the expected value function of the underlying dynamic problem, we use Gaussian quadrature and interpolation over an adaptively refined grid, and solve a potentially large non-linear system. Second, to evaluate the likelihood function, we decompose the integral over the unobserved state variables in the likelihood function into a series of lower dimensional integrals, and successively approximate them using Gaussian quadrature rules; we call this procedure recursive likelihood function integration (RLI). Finally, we solve the maximum likelihood problem using a nested fixed point algorithm.

First, we verify the RLI algorithm's ability to recover the parameters in an extensive Monte Carlo study with simulated data sets, finding that the method is indeed able to recover the parameters used for the simulation, particularly in the case of the serial correlation parameter, which is recovered to very high precision. Also, we find some evidence that the distribution of

the estimates is asymptotically normal for big enough data sets. Then, we apply this method to the bus engine replacement model of Rust (1987), and find significant serial correlation for some of the subsamples. Also, the parameter estimates vary substantially, compared to the case of serially uncorrelated errors. We wish to emphasize again that the method presented in this paper is not limited to the bus engine replacement model, but is generic with respect to DDCMs with Markov serial dependence in the unobserved state variables.

As we mention in the introductory section, the recursive likelihood function integration is not the only approach to the estimation of DDCMs with serially correlated unobserved state variables. While we cited some recent alternative methods, we did not compare to them in terms of runtimes, accuracy, or other important metrics. Rather, the goal of this paper was to show that the integration of the serially correlated variables in the computation of the likelihood function can be done with complexity that is linear in the time horizon, making the application of high performance quadrature rules such as Gaussian quadrature well feasible. For a quantitive comparison of the various methods to be insightful, a rigorous experimental design is needed, in order to compare the different aspects of computational efficiency, numerical accuracy, and scaling properties, based on unified models and environments. This in-depth comparison study is subject to future research.

	$EV1$						
	True	$M = 100$		$M = 1,000$		$M = 10,000$	
RC	26.0000	11.1135	34.8033	11.0589	32.8574	11.0402	27.7607
		(0.974)	(21.695)	(0.266)	(17.391)	(0.092)	(5.457)
θ_1	7.0000	2.9381	10.2193	2.9023	9.3360	2.9035	7.5885
		(0.435)	(7.381)	(0.121)	(5.789)	(0.041)	(1.748)
RC/θ_1	3.7143	3.8182	3.6105	3.8136	3.6508	3.8027	3.6809
		(0.275)	(0.372)	(0.089)	(0.236)	(0.028)	(0.103)
ρ	0.7000	–	0.6345	–	0.7077	–	0.7069
			(0.241)		(0.081)		(0.035)
p (LR)			0.2380		0.0003		$< 10^{-16}$
			(0.291)		(0.002)		$(< 10^{-16})$
	$N(0,1)$						
RC	18.0000	7.3097	20.3922	7.1433	20.5688	7.1349	19.8626
		(0.608)	(15.928)	(0.177)	(12.954)	(0.056)	(6.118)
θ_1	5.0000	1.8410	6.1094	1.7589	5.9685	1.7584	5.6418
		(0.268)	(5.391)	(0.077)	(4.295)	(0.025)	(1.990)
RC/θ_1	3.6000	4.0086	3.6162	4.0650	3.6236	4.0581	3.5676
		(0.291)	(0.403)	(0.095)	(0.282)	(0.031)	(0.143)
ρ	0.6000	–	0.4069	–	0.5243	–	0.6035
			(0.317)		(0.222)		(0.091)
p (LR)			0.6946		0.3302		0.0082
			(0.303)		(0.322)		(0.036)

Table 1.3: Mean and standard deviations of the maximum likelihood estimates from 200 artificial data sets of different sizes, density $q(\cdot)$ being extreme value type 1 $EV1$ (top), and standard normal $N(0,1)$ (bottom). The lefthand column uses data sets of 100 buses each, the center column 1,000 buses, and the righthand column 10,000 buses. L is the value of the log-likelihood function at the solution; p (LR) is the mean and the standard deviation of the p-values of the likelihood ratio test with $H_0 : \rho = 0$ carried out on the individual data sets; $\beta = .9999$.

	$EV1$					
	Bus groups 1–3		Bus groups 1–4		Bus group 4	
RC	11.7266	25.0029	9.7560	27.0159	10.0749	22.0389
	(1.928)	(38.239)	(0.898)	(16.929)	(1.351)	(33.115)
θ_1	4.8257	9.8452	2.6276	7.4199	2.2929	4.8137
	(1.366)	(17.501)	(0.469)	(5.567)	(0.554)	(8.546)
RC/θ_1	2.4300	2.5396	3.7128	3.6410	4.3935	4.5784
ρ	–	0.6896	–	0.7396	–	0.7000
		(0.335)		(0.112)		(0.329)
L	-2,708.366	-2,707.764	-6,055.250	-6,053.340	-3,304.156	-3,303.913
p (LR)		0.2724		0.0506		0.4863
	$N(0,1)$					
RC	7.0372	13.5964	6.0018	18.6650	6.0747	11.0085
	(1.029)	(23.726)	(0.481)	(7.653)	(0.758)	(34.280)
θ_1	2.5406	5.2870	1.3990	5.1993	1.1829	2.3233
	(0.732)	(10.516)	(0.263)	(2.488)	(0.327)	(8.425)
RC/θ_1	2.7700	2.5717	4.2900	3.5899	5.1354	4.7383
ρ	–	0.5143	–	0.6656	–	0.4945
		(0.682)		(0.103)		(1.382)
L	-2,707.901	-2,707.820	-6,054.082	-6,053.684	-3,303.919	-3,303.899
p (LR)		0.6862		0.3725		0.8446

Table 1.4: Estimation results for different subsamples of the original dataset, density $q(\cdot)$ being extreme value type 1 $EV1$ (top), and standard normal $N(0,1)$ (bottom). L is the value of the log-likelihood function at the solution; p (LR) is the p-value of the likelihood ratio test with $H_0 : \rho = 0$; $\beta = .9999$.

1.A Open Source Software

This appendix lists all open source software packages used to obtain the results presented in this paper, including version information.

The main framework used to implement the method of this paper is R, version 3.0.3 (R Core Team, 2014). Time critical components are implemented in C++, and interfaced to R using the "Rcpp" package, v0.11.1 (Eddelbuettel and François, 2011). The code is parallelized on the C++ level using openMP. All interpolation on the C++ level is carried out using the respective routines of the GNU Scientific Library, v1.16 (Galassi, Davies, Theiler, Gough, Jungman, Alken, Booth, and Rossi, 2014). Gaussian quadrature nodes are computed using the R-packages "fastGHQuad", v0.1-1 (Blocker, 2011), and "pracma", v1.6.4 (Borchers, 2014). Distribution functions, quantile functions, and random number generators for the extreme value distribution are provided by the R-package "evd", v2.3-0 (Stephenson, 2002). To numerically solve the fixed point problem (1.9), we use the "ipopt" package, v3.11.7 (Wächter and Biegler, 2005), in conjuction with the "pardiso" sparse linear solver, v5.0.0 (Schenk and Gärtner, 2004), interfaced by the R-package "ipoptr", v0.8.4, by Jelmer Ypma (which is distributed as part of the ipopt package), and the quasi-Newton trust-region method of the R-package "nleqslv", v2.1.1 (Hasselman, 2014). For the likelihood maximization problem, we employ "bobyqa" (Powell, 2009), interfaced by the "minqa" R-package, v1.2.3 (Bates, Mullen, Nash, and Varadhan, 2012).

Essay 2

MPEC Estimation with Flexible Grids

Adaptive Grids for Estimation of Dynamic Models using Constrained Optimization Approaches[1]

Gregor Reich
Dept. of Business Administration
University of Zurich
Moussonstrasse 15
8044 Zurich, Switzerland
gregor.reich@business.uzh.ch

Ole Wilms
Dept. of Business Administration
University of Zurich
Moussonstrasse 15
8044 Zurich, Switzerland
ole.wilms@business.uzh.ch

November 2014

Abstract

This paper develops a method to flexibly adapt interpolation grids of value function approximations in the estimation of dynamic models by MPEC (Su and Judd, 2012). Since MPEC requires the grid structure for the value function approximation to be hard-coded into the constraints, one cannot apply iterative node insertion for grid refinement; rather, we show how to adapt the grid by moving the nodes, a technique called r-adaption. We demonstrate how to obtain optimal grids based on the balanced error principle, and implement this approach by additional constraints to the likelihood maximization problem. The method is applied to the bus engine replacement model (Rust, 1987), modified to feature a continuous mileage state.

Keywords: Numerical dynamic programming; Mathematical programming with equilibrium constraints; r-adaptive grid refinement; Equioscillation.

[1]We are heavily indebted to our advisor Karl Schmedders, and to Ken Judd and Che-Lin Su for their support of this project. We also thank Philipp Renner, Simon Scheidegger, and participants of "Institute on Computational Economics, 2014" at Hoover Institution, Stanford, for helpful comments. Reich gratefully acknowledges financial support by Forschungskredit of the University of Zurich under grant no. K-33142-02.

2.1 Introduction

This paper develops a method to flexibly adapt interpolation grids of value function approximation in dynamic models, such as dynamic discrete choice models, when the estimation is done using constrained optimization, namely the MPEC approach of Su and Judd (2012). While being more efficient than the classical nested fixed point (NFXP) for some problems, the MPEC approach needs the structure of the value function approximation to be hard-coded into the constraints of the likelihood optimization problem. As a consequence, one cannot use iteratively adaptive procedures for grid refinement in every iteration of the optimization, as it is possible (and has been reported to be numerically favourable) in NFXP estimation. In this paper, we show how to adapt the interpolation grid by moving the nodes, a technique called r-adaptive refinement. We demonstrate how to obtain optimal grids (given a fixed number of nodes), and show how to integrate this approach into the likelihood maximization problem using the equioscillation principle. The method is applied to the bus engine replacement model of Rust (1987), modified to feature a continuous mileage state.

Many models in modern applications of structural estimation assume the agents to behave dynamically optimal. A popular example is the literature on dynamic discrete choice models (DDCM), pioneered by the seminal work of Rust (1987, 1988); for recent surveys on the estimation of DDCMs, see Aguirregabiria and Mira (2010); Keane et al. (2011); Arcidiacono and Ellickson (2011). In these applications, data on the decisions of one or more agents in a particular dynamic problem are observed, along with other state variables that enter the agents' optimization. Using this data, the structural parameters of the model such as parameters of the utility functions or of the law of motion of the state variables are estimated, for example by the method of maximum likelihood or Bayesian approaches.

The estimation of dynamic models is challenging both methodologically and computationally, as in principle, the dynamic optimization problem and the likelihood maximization problem have to be solved simultaneously. While there exist methods that avoid solving the model by estimating the conditional choice probabilities directly from the data going back to the work of Hotz and Miller (1993), and methods that avoid to explicitly maximize the likelihood by using Bayesian approaches with Markov chain Monte Carlo simulation (Imai, Jain, and Ching, 2009; Norets, 2009), many of the widely used workhorse algorithms still rely on solving both problems simultaneously, as they have shown excellent numerical and statistical efficiency.

In particular, the nested fixed point (NFXP) algorithm of Rust (1987) and the constrained optimization approach (mathematical programming with equilibrium constraints, MPEC) of Su and Judd (2012) are among the most used algorithms. In NFXP, the dynamic problem is solved iteratively within an "inner loop", and its solution is used to obtain the choice probability to compute the likelihood function for a particular parameter value, which is itself maximized in an "outer loop". The methodological separation of the two steps makes this algorithm a robust and efficient choice for many applications, as the very special structure of the two problems can be exploited; in particular, this includes the contraction mapping property of the

underlying dynamic problem, and structure of likelihood maximization problems, where the Hessian can be computed as the outer product of the gradients. On the other hand, the MPEC approach tries to avoid computing the full solution of the dynamic problem throughout the likelihood optimization, except at the optimal parameter vector itself. This can potentially be achieved by inserting optimality conditions of the underlying dynamic problem as non-linear constraints into the likelihood maximization, and solving the resulting problem using constrained optimization techniques. The fact that the solution of the model is not computed at every iteration of the optimization is conceptually very attractive, as it often makes up for most of the computation time, but solutions other than at the optimal parameter vector rarely have any relevance. Therefore, MPEC has shown excellent numerical efficiency in many applications.

When solving dynamic models with continuous state spaces, most methods to approximate the value function require the researcher to specify a grid over the domain of approximation. Depending on the approximation scheme in use, the nodes of this grid serve as interpolation or collocation points, and potentially as break points if the approximation is assembled from piecewise basis functions. Popular grids are the uniform grid and the Chebyshev grid. While these choices might be good "ex ante" without much knowledge about the approximated function, they are generally suboptimal once the function – or an approximation thereof – is known. Consequently, a popular approach to grid creation is iterative refinement: given the grid from the last iteration (or starting from a uniform grid), the unknown function is approximated, and based on some approximation error criterion, a new grid is created by inserting additional nodes in regions of high approximation error; this procedure is repeated until the maximum approximation error is below some threshold. Iterative grid refinement methods have successfully been applied to dynamic programming problems with continuous state variables in economics; see, for example, Grüne and Semmler (2004); Brumm and Scheidegger (2013); Reich (2014).

As we will demonstrate, iterative refinement can well be integrated into NFXP, as it is agnostic about the function approximation algorithm used in model solution step. However, the integration of iterative refinement with MPEC is not obvious for two reasons: First, inserting an interpolation or collocation node into the grid corresponds to inserting a constraint into the likelihood optimization problem while the optimizer runs, which is generally unstable. Second, since MPEC gives no guarantee that the dynamic problem is solved at any point of the optimization except for the solution, no valid value function approximation exists on which a criterion of where to actually insert nodes can be evaluated. Therefore, the integration of MPEC with flexible grids calls for a different methodology.

An alternative to grid refinement by node insertion is adaption by node movement: Instead of inserting a new node in a region of high approximation error, an existing node is moved there from a region of low approximation error, keeping the total number of nodes in the grid fixed. Obviously, finding a good (or optimal) grid is more difficult, as moving a node does not only affect the approximation in the region where the node is moved to, but also where it has been moved away. While not being as popular as the iterative schemes, grid adaption

by node movement has attracted attention for example in the literature on free-knot splines for curve fitting (see, for example, Schumaker, 2007, and the literature cited therein), and in the solution of partial differential equations with moving meshes (see, for example, Huang and Russell, 2011, and the literature cited therein).

A particular difference between node insertion and node movement is the adaption criterion. In the case of insertion, this criterion is usually binary, as nodes are inserted at pre-specified locations if the approximation error locally exceeds some threshold, otherwise not. In contrast, the node movement is a continuous operation, as for each node, the position on the refined grid (or the direction of movement) must be specified; these are either obtained from conditions on the node positions based on the approximated function (for example equidistributed over the function's values or over its arc-length), or from the solution of a minimization problem over the approximation error; see Baines (1998) for a comparison of the two criteria.

The method we develop in this paper is based on approximation error minimization. However, directly integrating this optimization problem into MPEC results in a bi-level optimization problem, where the likelihood maximization is "wrapped around" the approximation error minimization, which makes the resulting combined problem difficult to solve. A common approach to bi-level optimization is to replace the lower-level problem by its first-order optimality constraints – in the case of inequality constraint lower-level problems the Karush-Kuhn-Tucker (KKT) conditions – and solve the resulting problem using constrained optimization techniques. However, solving bi-level problems using first-order necessary conditions is known to cause numerical issues, such as difficult to handle complementarity constraints which violate the constraint qualifications of the first-order conditions of the combined problem; for a discussion of this issue, see, for example, Colson, Marcotte, and Savard (2007); Fletcher, Leyffer, Ralph, and Scholtes (2006).

Nevertheless, our method also replaces the lower-level approximation minimization problem by a set of optimality constraints using (in)equalities only, and thus allow the problem to be solved by standard constrained optimization solvers. However, instead of relying on generic conditions such as KKT, the structure of our problem makes it possible to use a classical result from numerical analysis to derive an alternative set of conditions, that is actually even sufficient for optimality. The "equioscillation" theorem for polynomial approximation and its generalizations to other functional forms state that if an approximation of a continuous function has "balanced and alternating errors", it is uniform (best approximation in the L_∞ norm); geometrically speaking, best approximations have errors that oscillate with an amplitude equal to the maximum absolute error.

Using the balanced error (BE) principle, we show that it is straightforward to derive a set of (in)equalities that form optimality conditions for the approximation error minimization, which we finally integrate into the MPEC problem, and which we solve simultaneously with the likelihood maximization problem using standard constrained optimization. In summary, we derive a procedure to integrate grid adaption by node movement into estimation of dynamic

models using constrained optimization (MPEC), resulting in value function approximations that are optimal in the L_∞ norm, given their functional form and the total number of grid nodes.

In the second part of this paper, we present a series of examples and applications in order to verify our method, and demonstrate its mechanics and performance advantages.

In Section 2.3.1, we apply approximation with BE conditions to various standard function interpolation examples, leaving aside the MPEC estimation of any model. We do this in order further illustrate the principle of equioscillation and its application to grid adaption, and, moreover, to verify our method by putting it in relation to other approaches and theoretical results: First, all function approximation examples are computed both using BE conditions, and by direct minimization of the approximation error measured by the L_∞ norm; like this, we also experimentally confirm the equivalence of the two problems. Interestingly, while we find the solutions to be identical as expected, the BE variant appears to be numerically much more efficient than direct minimization. Second, we relate our approach to a theoretical result on static node choice, the Chebyshev nodes: As we argue in Detail in Appendix 2.A.2, a grid composed from Chebyshev nodes cannot be better than a grid obtained from equioscillation. However, we present a (non-trivial) numerical example which is constructed such that the two solutions must coincide; consequently, we can provide a closed form solution benchmark on which to verify our method and its implementation.

In our experiments, we approximate different polynomials and the exponential function, using polynomial interpolation on Chebyshev nodes, and piecewise linear and piecewise quadratic interpolation on uniform grids, and compare their accuracy to the respective approximations over flexible grids obtained from BE conditions. We find that in all cases, the flexible grid allows for more accurate approximation given the same number of interpolation nodes (except for the closed form benchmark, which has identical accuracy).

In Section 2.3.2, we apply our method to the bus engine replacement model of Rust (1987), which we modify to feature a continuous mileage state.[2] Consequently, the computation deviates from the original model in two ways: First, we have to specify a continuous mileage transition process, and second, the expectation over the one period ahead values in the Bellman equation is continuous and thus has to be approximated using numerical quadrature. Similar to the function approximation examples, we first approximate the expected value function for a fixed parameter vector; we do this in order to verify the method by comparing it to a benchmark solution, which is computed using a very fine grid, as well as to demonstrate the potential efficiency gains from node movement in a particular application. We find that even in this simplified problem, our method uses significantly fewer nodes to attain a pre-specified level of accuracy, compared to a uniform grid.

[2]Similarly, Kristensen and Schjerning (2014) estimate the bus engine replacement model of Rust (1987) with continuous mileage; their aim is to quantify approximation errors from various sources such as the discretization of naturally continuous state variables.

Finally, we estimate the full model using constrained optimization (MPEC) with a flexible grid. In order to obtain a measure of variation of the estimates and the corresponding errors, we carry out a Monte Carlo study with 100 artificial datasets. We find that the comparative advantage compared to a uniform grid in terms of accuracy and computational efficiency is substantial, even in this simple one-dimensional application: The mean squared error of a uniform grid is an order of magnitude higher compared to a grid of equally many flexible nodes; conversely, estimating the model using a uniform grid that is fine enough to achieve the same accuracy as its flexible counterpart results in roughly 1.5 times higher computation times.

The remainder of this paper is structured as follows: Section 2.2 will motivate the problem of estimating dynamic models and present solution algorithms, motivate the use of grid adaption and introduce different approaches, derive sufficient conditions for uniform approximation, and finally integrate the conditions with the MPEC estimation procedure. In order to both verify and illustrate our method, Section 2.3 first applies the uniform approximation conditions to standard interpolation problems as well as to the solution of the dynamic model of Rust (1987) for fixed parameter vectors, and finally applies the method to the full parameter estimation problem. Section 2.4 concludes and states the agenda for future research. Appendix 2.A gives a very short introduction to function approximation using polynomial approximation, piecewise polynomial approximation, and splines, mainly to ensure precise nomenclature throughout the paper.

Note that the method presented in this paper is currently limited to value functions of one continuous state variable only; the generalization to higher dimensions is subject of further research.

2.2 MPEC Estimation with Flexible Grids

This part of the paper develops a method for the use of flexible grids within the constrained optimization approach to the maximum likelihood estimation of dynamic models. To keep the discussion of our method generic, we will not specify a concrete type of model or application in this part of the paper; in the second part, we will demonstrate the method by applying it to the bus engine replacement model of Rust (1987), which is a well known model from the dynamic discrete choice literature.

2.2.1 Estimation of Dynamic Models

In this subsection, we will state the formal problem of estimating dynamic models by maximum likelihood, present methods to solve the dynamic problem and the likelihood maximization, and motivate the use of adaptive grid methods in this context.

2.2.1.1 Problem Statement

We begin with the formal statement of the problem we attempt to solve. Consider the following discrete time, infinite horizon, continuous (or mixed discrete-continuous) state dynamic optimization problem

$$V_\theta(x_0) = \max_{\{U_t(x_t)\}_{t=0}^{\infty}} \mathbb{E}\Big[\sum_{t=0}^{\infty} \beta^t \pi(x_t, y_t; \theta)\Big] \quad \text{s.t. } y_t \in D(x_t) \tag{DP}$$

where y is the control variable, which is at all times required to be in the feasible set of controls $D \subseteq \mathcal{D}$, given state $x \in \mathcal{S} \equiv \bar{\mathcal{S}} \times \tilde{\mathcal{S}}$; U is a policy function that maps every state x to a dynamically optimal response y; π is the instantaneous pay-off function; $\beta < 1$ is the discount factor; the distribution over future values of the state x' conditional on current values of state and control is given by $Pr(x'|x, y; \theta)$, which we also call the law of motion of the state variables; θ is an m-dimensional real valued parameter vector acting on the pay-off function $\pi \in$ and the law of motion $Pr(x'|x, y; \theta)$. (We assume the problem to be time stationary, and thus can drop all time indices.)

As shown by Bellman (1952), the following functional equation constitutes a necessary optimality condition to problem (DP):

$$V_\theta(x) = \max_{y \in D(x)} \{\pi(x, y) + \beta \mathbb{E}[V_\theta(x')|x, y]\} \equiv T[V](x) \tag{2.1}$$

where $V_\theta : \mathcal{S} \to \mathbb{R}$ is called the value function, which implicitly depends on θ through the pay-off function and the law of motion. In the following, we will address the dynamic problem solely in terms of its Bellman equation (2.1).

Suppose we do not only want to solve (DP), but rather, given a dataset consisting of (partial) data on state and control realizations $\{\tilde{x}_t, y_t\}_{t=1}^T$, $\tilde{x} \in \tilde{\mathcal{S}}$, we want to identify the parameter, θ, of the pay-off function π and the law of motion of states $Pr(x'|x, y; \theta)$, that maximizes the likelihood of the data, given the dynamic problem is solved at the solution of the maximum likelihood estimation (MLE) problem. This problem frequently arises in different fields of econometrics, for example in dynamic discrete choice modelling (DDCM); see, for example, Aguirregabiria and Mira (2010); Keane et al. (2011); Arcidiacono and Ellickson (2011) for surveys on DDCMs and their estimation. Formally, we attempt to solve the following two problems simultaneously

$$\left.\begin{aligned} \theta^* &= \arg\max_\theta \; L(\theta; V_\theta, \{\tilde{x}_t, y_t\}_{t=1}^T) \equiv Pr\big(\{\tilde{x}_t, y_t\}_{t=1}^T; \theta, V_\theta\big) \\ V_\theta(x) &= \max_{y \in D(x)} \{\pi(x, y; \theta) + \beta \mathbb{E}[V_\theta(x')|x, y; \theta]\}\Big|_{\theta=\theta^*} \end{aligned}\right\} \tag{MLDP}$$

The two problems are connected in the following way: The link between likelihood function and the model is through the parameter vector, which enters the pay-off and the law of motion of the states. In the other direction, the value function for a given parameter value enters the

likelihood function through the probabilities or density functions for the choice variables and the observable states in the data, $Pr\big(\{\tilde{x}_t, y_t\}_{t=1}^T; \theta, V_\theta\big)$, out of which the likelihood function is composed. Variables of the model for which no data is observed will be integrated out in the likelihood computation.

2.2.1.2 The Projection Method to Solve (DP)

We now turn to the description of solution techniques for the standalone dynamic problem in terms of its Bellman equation (2.1), which is a topic well covered by the literature (see, for example, Cai and Judd, 2013; Judd, 1998; Rust, 1996). A popular family of solution methods are the projection methods, out of which we choose the collocation method; see Judd (1992, 1998). The reason for this particular choice will become apparent at a later stage, when deriving a set of optimality conditions for the value function approximation. We now briefly describe the collocation method and some technical details, but only to the extent necessary for the derivation of our adaptive grid method.

Finding a function $V(\cdot)$ that solves the functional equation (2.1) is an infinite-dimensional problem, as functions are generally infinite-dimensional objects (even if the domain of the function is finite-dimensional). However, many functions can be approximated well by (sums of) basis functions with finite-dimensional representations.[3] For example, approximations using polynomials of finite degree can be represented by a finite-dimensional vector of coefficients. Projection methods replace the true function $V(\cdot)$ in equation (2.1) by its approximation $\hat{V}(\cdot; \mathbf{a})$ parametrized by a vector $\mathbf{a}$. Obviously, the Bellman equation will only be approximately satisfied, and the different kinds of projection methods are all ways to "make this error small". Formally, we define the residual as

$$R_{\hat{V}}(x; \mathbf{a}) \equiv \hat{V}(x; \mathbf{a}) - T[\hat{V}](x; \mathbf{a}) \tag{2.2}$$

The projection methods differ in the way they project the residual function against different test functions (including the residual itself, which results in a least squares approximation); these projections are then minimized or set equal to zero. The collocation method ensures that the residual function is zero at a chosen vector of n collocation nodes, $\mathbf{x} \in \mathcal{S}^n$. Note that this is equivalent to interpolation, if the function to be approximated can be evaluated directly. In order to solve the dynamic problem using collocation together with polynomial approximation of degree $n-1$, we need n nodes to identify the coefficients of the polynomial (or the "degrees of freedom", as they are often referred to in the literature), and solve a (generally non-linear) system of equations with the coefficients being the variables, and one equation for each collocation node (using more/fewer collocation nodes will result in an over-/under-

[3] We give a very short introduction to function approximation using polynomial approximation, piecewise polynomial approximation, and splines in the appendix, mainly to ensure precise nomenclature throughout the paper.

identified system of equations, with generally no/infinitely many solutions, respectively):

$$R_{\hat{V}}(x_i; \mathbf{a}) = 0, \ \forall x_i \in \mathbf{x} \tag{CO}$$

If the value function is approximated using piecewise polynomial approximation or splines, a vector of breakpoints has to be chosen as well. In the case of piecewise linear approximation, or splines of any order, the number of breakpoints equals the number of collocation nodes, and it is natural to choose them to be identical. However, if higher-order piecewise polynomial approximation is used without imposing additional smoothness constraints, more collocation nodes are needed in order to identify the degrees of freedom. In the remainder of this paper, we use approximation methods that either allow us to treat the collocation nodes and the breakpoints as one single set of nodes, or we simply distribute the additional collocation nodes uniformly between the breakpoints without making it explicit in the collocation equation (CO), to not complicate the notation without adding any more insight into our approach.

2.2.1.3 Maximum Likelihood Estimation of Dynamic Models

Having argued that the infinite time horizon, continuous state dynamic program (DP) can be represented and solved approximately as a non-linear system of equations (CO), we now turn to the solution of the full estimation problem (MLDP). We will describe the two most popular approaches to the estimation of dynamic models, the nested fixed point (NFXP) approach of Rust (1987), and the constrained optimization approach (or mathematical programming with equilibrium constraints, MPEC) approach by Su and Judd (2012); we proceed in chronological order.

The nested fixed point algorithm of Rust (1987) addresses problem (MLDP) by completely solving the dynamic problem not only at the maximum of the likelihood function, but for every guess of the parameter vector θ, as a repeated two step procedure; see Algorithm 2.1. It is important to note that the evaluation of the likelihood function is completely agnostic

Algorithm 2.1 Nested fixed point algorithm (Rust, 1987)

1: initialize θ, $\mathbf{a}$, $\mathbf{x}$
2: **while** θ not optimal **do**
3: find $\hat{V}_\theta(\cdot, \mathbf{a})$ such that $R_{\hat{V}_\theta}(x_i; \mathbf{a}) = 0, \ \forall x_i \in \mathbf{x}$
4: evaluate $L(\theta; \hat{V}_\theta(\cdot, \mathbf{a}), \{\tilde{x}_t, y_t\}_{t=1}^T)$ and compute next θ
5: **end while**

about the value function approximation step, as long as it is provided a function that can be evaluated over the whole state space. This not only allows for a wide variety of algorithms (and combinations thereof) to solve the dynamic problem, but also to make use of important properties of the problem, such as the contraction mapping property; moreover, the fact that the function to be maximized is a likelihood function can be exploited, for example when

computing its Hessian matrix.

In contrast, the MPEC approach of Su and Judd (2012) interprets problem (MLDP) as a bi-level optimization problem, where the lower-level problem is replaced by some optimality (or equilibrium) constraints, in our case the Bellman equation (2.1), hence its name. As argued above, the Bellman equation has to be replaced by a finite-dimensional approximation, in our case the collocation system (CO), yielding

$$\max_{\theta,\mathbf{a}} L(\theta; \hat{V}_\theta(\cdot, \mathbf{a}), \{\tilde{x}_t, y_t\}_{t=1}^T) \tag{2.3a}$$

$$\text{s.t. } R_{\hat{V}_\theta}(x_i; \mathbf{a}) = 0, \ \forall x_i \in \mathbf{x}. \tag{CO}$$

Conceptually, the motivation for MPEC is to avoid solving the dynamic problem for every parameter value on the trajectory of the likelihood maximization, but rather increase feasibility (here: accuracy of the solution to the dynamic problem) and optimality (here: likelihood value) simultaneously. Of course, since the solution of the dynamic problem depends on the actual value of the parameter vector, this is a potentially difficult non-linear constrained optimization problem.

Whether to use NFXP or MPEC to estimate the parameters of dynamic models using MLE is an active field of research, and might turn out to be highly problem dependent. While it is argued that MPEC might be more efficient because it does not require to solve the dynamic model in each iteration, it clearly cannot make use of the contraction mapping property as it is done within NFXP, which might cause MPEC to use more iterations to solve the likelihood problem (or even fail to converge). Also, the memory needs are very different, as both algorithms can use sparsity of the problem in the Jacobian of the collocation system (if present), but the Hessian of the MPEC problem is much larger than the one in NFXP, and is, moreover, generally not sparse. In this paper, we make no attempt to contribute any evidence in favour of either MPEC or NFXP. Rather, we assume that the researcher we address has made his choice for MPEC and is looking for a way to make the grid creation more efficient. However, to better motivate our approach, we discuss a particular difference between the two approaches in the following.

Another difference between MPEC and NFXP is the way approximations of continuous value functions are handled: In NFXP as defined in Algorithm 2.1, the procedure of obtaining an approximation to the value function is technically independent from the likelihood maximization, as long as the optimizer is fed with a valid approximation of the value function (given a particular value for the parameter vector) in order to evaluate the likelihood. In particular, this algorithm is suitable for the application of any iterative refinement scheme for the grid over state variables of the dynamic problem. Algorithm 2.2 conceptually extends NFXP with iterative grid update (for every parameter value), as successfully applied by Reich (2014).

In contrast to NFXP, the application of grid adaption techniques is not obvious in the MPEC

Algorithm 2.2 Nested fixed point algorithm with iterative grid refinement (Reich, 2014)

initialize θ, $\mathbf{a}$
while θ not optimal **do**
    initialize $\mathbf{x}$
    while $\|V_\theta - \hat{V}_\theta\| > \eta$ **do**
        find $\hat{V}_\theta(\cdot, \mathbf{a})$ such that $R_{\hat{V}_\theta}(x_i; \mathbf{a}) = 0$, $\forall x_i \in \mathbf{x}$
        refine $\mathbf{x}$
    end while
    evaluate $L(\theta; \hat{V}_\theta(\cdot, \mathbf{a}), \{\tilde{x}_t, y_t\}_{t=1}^T)$ and compute next θ
end while

approach: If, for example, the value function is obtained by projection using collocation, for each collocation node the corresponding equality constraint has to be specified in the MPEC problem. Iterative refinement by insertion (or deletion) of nodes in the way it is done in Algorithm 2.2 is not directly applicable for two reasons: First, inserting a collocation node corresponds to inserting a constraint, which generally cannot be done while the optimization runs;[4] second, since MPEC gives no guarantee that the dynamic problem is solved at any point of the optimization except for the solution to the MLE problem, no straightforward criterion of where to actually insert or delete nodes can be derived. Consequently, the application of grid adaption to MPEC raises the need for grid adaption schemes other than the popular iterative methods, as the structure of the function approximation problem is "hard-coded" into the optimization problem.

2.2.1.4 Types of Grid Adaption

As we have pointed out, applying grid refinement techniques within the NFXP framework is straightforward, because the likelihood maximization is agnostic of the value function approximation algorithm. However, integrating grid adaption with MPEC is more involved. To better motivate our approach to MPEC with grid adaption, we first give an overview over the different concepts of grid adaption, and argue why they might be suited for our purpose, or why they are not.

The refinement of function approximations is a well-studied problem in the literature, and has been successfully applied to the solution of dynamic problems with continuous state variables in economics (see, for example, Grüne and Semmler, 2004; Brumm and Scheidegger, 2013; Reich, 2014). Following Huang and Russell (2011), we classify these methods as follows:

h - adaptive refinement: The most popular refinement method is adaption by node inser-

[4] One way to "insert" or "remove" constraints while the optimizer runs are the so-called on/off constraints, where an additional binary variable is multiplied against the corresponding constraints. However, such an approach would increase the complexity of the optimization problem significantly, and is not pursued in the paper.

tion: Based on some criterion such as the residual function, additional nodes are inserted at places where the approximation quality is "poor" (or deleted at locations where they are not needed), and the approximation problem is re-solved using the refined grid, with the interpolation or collocation conditions (CO) being enforced also at the new nodes. The resulting grid data structure usually forms a hierarchy of grids of different refinement levels. While this approach is intuitive and relatively easy to implement, it is intrinsically iterative in the sense that it needs a temporary solution of the problem in order the compute the next, refined solution; as already pointed out, this rules out its integration into MPEC.

p **- adaptive refinement:** If the degree of curvature of the approximated function is substantially varying over the domain, it is often efficient to locally increase the degree of the approximating polynomials, in order to better capture these features. The additional degrees of freedom can be identified by inserting more grid nodes in the interior of the grid cells in question. Consequently, this method cannot be integrated with MPEC either, for the same reasons as for the h-adaption.

r **- adaptive refinement:** In contrast to h- and p-adaption, r-adaption does not change the structure of a particular grid, but rather moves its nodes such that local features of the approximated function are well covered. Consequently, the total number of nodes, as well as their (dimension-wise) ordering is preserved. Formally, the coordinate transformation is a monotone mapping between a canonical grid over the unit hypercube ("computational domain"), and the adapted grid defined over the domain of approximation ("physical domain"). This mapping can either be explicitly known from the physical problem, approximated before solving the actual problem, or it is solved for simultaneously with the function approximation problem itself. In all cases, a popular criterion is equi-distribution of nodes, which uniformly distributes nodes for example according to the gradient of the approximated function, or on its arc-length. Alternatively, if the function of interest (or an approximation thereof) can be evaluated, a direct minimization problem over the approximation error can be solved; see Baines (1998) for an in-depth comparison of the different criteria.[5] Since the r-adaption of a grid is done without changing functional form of the approximation, this idea can potentially be integrated with MPEC.

In this paper, we develop a method with r-adaptive grids that is based on approximation errors rather than equi-distribution for two reasons: First, equi-distribution generally requires that either the gradient of the approximated function can be evaluated as well, or its arc-length can be computed, which is not always possible in value function approximation; second, depending on the type of interpolation, placing nodes at regions of high curvature might be even more accurate compared to placing them in regions of steep gradients.

[5] More recently, attempts have been made to unify the two approaches, in the sense that the approximation error directly enters the criterion function on which equi-distribution is imposed, the so-called monitor function; in particular, see Huang and Russell (2011). However, the theoretical results on convergence towards approximation error minimizing solutions are still missing.

In summary, we have stated the problem of estimating dynamic models using maximum likelihood, presented the collocation method to solve the standalone dynamic problem, and shown how two popular approaches, namely NFXP and MPEC, integrate collocation to estimate the model. Furthermore, we have argued that while it is easy to integrate different types of grid adaption for the value function of the dynamic problem with NFXP, it is not obvious how to integrate it with MPEC, as the structure of the function approximation is hard-coded into the optimization problem. In the next section, we turn to uniform function approximation, and optimal breakpoint and collocation node distribution, which we finally show how to integrate with the MPEC algorithm.

2.2.2 Uniform Approximation and the Balanced Error Property

In this section, we briefly introduce the concepts of uniform approximation, equioscillation, and "balanced errors", which are then applied to form a criterion for node placement in r-adaption within MPEC in the next section.

2.2.2.1 Uniform Approximation and Node Placement

Recall that we defined the residual function (2.2) as the difference of the unknown function and its approximation; the closer we get the residual to zero over the whole domain, the better our approximation will be. A formalization of this is the uniform approximation problem, which minimizes the maximum absolute error between the unknown function and its finite-dimensional approximation with generic parameter vector $\mathbf{p}$:

$$\min_{\mathbf{p}} \|R_{\hat{V}}(x;\mathbf{p})\|_\infty \equiv \max_{x\in\mathcal{S}} |R_{\hat{V}}(x;\mathbf{p})|. \tag{2.4}$$

It is important to note that there is no explicit notion of nodes in problem (2.4) yet. (Consequently, there are no collocation constraints either.)

As we pointed out earlier, we assume the breakpoints of a piecewise polynomial approximation and the collocation nodes to coincide. Suppose we approximate the solution to the Bellman equation using one of these methods, and further suppose we interpret one part of vector $\mathbf{p} \equiv (\mathbf{x}, \mathbf{a})$ as nodes $\mathbf{x}$ that serve as both collocation and breakpoints, and the other part as the corresponding coefficients $\mathbf{a}$. While most approaches to function approximation assume the nodes to be fixed, we treat them as variables of the following uniform approximation problem:

$$\min_{\mathbf{a},\mathbf{x}} \|R_{\hat{V}}(x;\mathbf{a},\mathbf{x})\|_\infty \equiv \max_{i\in I} \max_{x_i\le x<x_{i+1}} |R_{\hat{V}}(x;\mathbf{a},\mathbf{x})| \tag{2.5a}$$

$$\text{s.t. } R_{\hat{V}}(x_i;\mathbf{a},\mathbf{x}) = 0,\ \forall x_i \in \mathbf{x} \tag{CO}$$

$$\mathbf{x} \in \mathcal{X}^n \tag{2.5b}$$

where $I = \{1, \ldots, n+1\}$ is the set of grid cell indices in one dimension.[6] Besides adding the node variables, we also added two sets of constraints: First, the grid validity conditions (2.5b) ensure that the structure of the grid such as neighbour ship relations are preserved; in the one-dimensional case, the set of all valid grids over the state space is defined by $\mathcal{X}^n \equiv \{\mathbf{x} \in \mathcal{S}^n : x_i \leq x_{i+1}\}$. Second, the collocation constraints (CO) are added. Note that none of these constraints is conceptually necessary at this point. However, we include them for a reason: The grid validity constraints rule out a great number of local solutions to the approximation problem with different orderings of nodes; adding the collocation constraint cannot improve the quality of the approximation problem, but is mandatory for our optimality criterion which we derive shortly to be applicable.

If the vector of breakpoints and collocation nodes is fixed, the coefficients of the approximation problem are identified solely by the constraints, and thus no explicit minimization is necessary (or possible). While we treat the nodes as variables in this paper, we still want to mention a popular approach of static node choice in the case of (non-piecewise) polynomial approximation, the Chebyshev nodes: Suppose that the function f we approximate is $k \geq 1$ times continuously differentiable. Suppose q_{n-1} is a polynomial interpolant of degree $n-1$ that interpolates f at the n roots of the degree n Chebyshev polynomial. Then, q_{n-1} minimizes the tightest known error bound for polynomial interpolation that can be minimized over the nodes *independently* of f (for a detailed description of Chebyshev nodes, see Appendix 2.A.2). While this is unquestionably a strong result with large practical implications, we show in numerical examples below that Chebyshev nodes are generally not optimal *given* a specific f. Also, Chebyshev nodes have no direct application in the context of breakpoint choice for the piecewise interpolation schemes, which are our methods of choice.

2.2.2.2 Equioscillation and Balanced Errors

While problem (2.5) can, in principle, be solved directly using non-linear constrained optimization techniques, we now re-formulate it as a set of optimality conditions, in order to easily integrate it as constraints with the original estimation problem using MPEC. (Moreover, as we demonstrate in the numerical part of the paper, the direct minimization approach is much less efficient in practice compared to directly solving the optimality conditions derived below.)

To derive our approach to optimal node placement, we restate an important result from polynomial approximation theory, the *equioscillation* theorem (restated from Judd, 1998, p.212): Define the L_∞ error of the best approximation of a (one-dimensional) function $f \in \mathcal{C}^k$, $k \geq 0$ by a polynomial of degree $n-1$ or less, $q^* \in \mathcal{P}^{n-1}$, as

$$\varrho_{n-1}(f) \equiv \inf_{\{q \in \mathcal{P}^{n-1} \,:\, deg(q) \leq n-1\}} \|f - q\|_\infty \tag{2.6}$$

[6] We assume the boundary of the grid to be fixed, and do not include it in the node count; this is w.l.o.g., but allows for a more consistent notation. Whether or not collocation is enforced at the boundary depends on the approximation scheme in use (enforced for piecewise polynomial, not enforced for polynomial approximation).

Then, we can state the following theorem:

Theorem 1 (equioscillation)**.** *If $f \in \mathcal{C}[a,b]$, then there is a unique polynomial of degree $n-1$, $q^*_{n-1}(x)$, such that $\|f - q^*_{n-1}\|_\infty = \varrho_{n-1}(f)$. The polynomial q^*_{n-1} is also the unique polynomial for which there are at least $n+1$ points $a \leq y_0 < \cdots < y_n \leq b$ such that for $m = 1$ or $m = -1$,*

$$f(y_j) - q^*_{n-1}(y_j) = m(-1)^j \varrho_{n-1}(f),\ j = 0, \ldots, n. \tag{2.7}$$

From the equioscillation theorem we know that the best polynomial approximation of the continuous function f will have "balanced and alternating errors". More precisely, if x_i, $i \in \{1, \ldots, k\}$ are the $k \geq n$ zeros of $f(x) - q(x)$, and $x_0 = a$, $x_{k+1} = b$, we call the errors to be balanced if

$$\max_{x_i \leq x \leq x_{i+1}} |f(x) - q(x)| = c,\ i = 0, \ldots, k, \tag{BE}$$

where c is a constant, and to alternate if, for $m = 1$ or $m = -1$,

$$\operatorname{sign}(f(x) - q(x)) = m(-1)^i,\ x_i < x < x_{i+1},\ i = 0, \ldots, k. \tag{2.8}$$

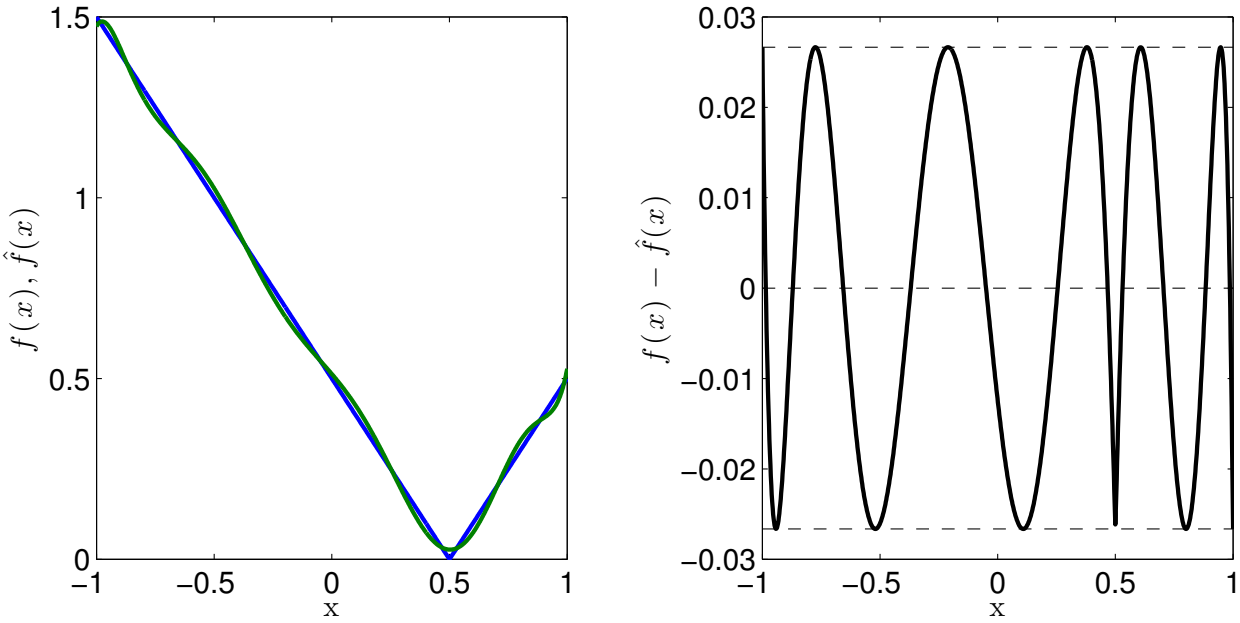
For an illustration, we include Figure 2.1 where the absolute value function is uniformly approximated by a polynomial, depicting both the function and its approximation the left panel, as well as the approximation error in the right panel; as predicted by the equioscillation theorem, the errors of the L_∞-minimizing polynomial approximation are balanced and n times alternating.

It is important to note that Theorem 1 states a sufficient condition for optimality: Since the *unique* best approximating polynomial of degree $n-1$ or less exists for every continuous function, and since it is also the *unique* polynomial with n times equioscillating errors, we can conclude that a polynomial with n times equioscillating errors is the best approximation of f (of degree $n-1$ or less) in the sense of definition (2.6). An iterative procedure to obtain best polynomial approximations of functions of one variable based on equioscillation is the Remez algorithm (see, for example, Fraser, 1965).

So far, we only considered polynomial approximation, where the interpolation or collocation nodes are determined implicitly by the zeros of $f(x) - q(x)$. However, when using piecewise polynomial approximation or splines, the choice of breakpoints (and possibly also interpolation nodes, if not identical) is explicitly required from the user. Moreover, the number of degrees of freedom (coefficients) of piecewise polynomials is generally a multiple of the comparable polynomial case — depending on the degree of the segments — even if the breakpoints are fixed.[7]

[7] For example, fitting a piecewise linear approximation with $n+1$ segments requires $2(n+1)$ parameters to be identified, compared to the n parameters of a degree $n-1$ best polynomial in the sense of Theorem 1, the residual of which oscillates at least $n+1$ times. (Note that at this point, we do not assume that the piecewise approximation satisfies any interpolation constraints, which would reduce the number of required parameters to $n+2$ in the piecewise linear case.)

Figure 2.1: Approximation of $f(x) = |x - 0.5|$ Using the Best Degree 10 Polynomial Approximation



Approximation of $f(x) = |x - 0.5|$ using the best degree 10 polynomial on the interval $[-1, 1]$. The blue line shows the true function $f(x)$ and the green line shows the best polynomial approximation $\hat{f}(x)$. The black line shows the approximation error $f(x) - \hat{f}(x)$.

Therefore, sufficient optimality conditions based on balanced errors have been derived for the specific piecewise schemes as well. In particular, Lawson (1964) proved for piecewise polynomials that if all segments are locally minimizing the maximum approximation error, then balancing these "min-max" errors over all segments is a sufficient condition for optimal breakpoint choice. Note that no alternation property is required in this case; also, all degrees of freedom are well identified, as the coefficients are determined by the segment-wise error minimization, and imposing balanced min-max errors over $n + 1$ segments implies additional $n + 1$ constraints for identifying the n positions of the flexible nodes.

For the case of splines, Schumaker (1968) showed that a result similar to Theorem 1 can be established. However, the properties of piecewise polynomials over flexible grids are well studied also for higher dimensions in the finite element literature (see, for example, Ciarlet, 2002), which is not generally true for splines (see Thompson, Soni, and Weatherill, 2010). As our goal is to eventually generalize our method to the case of higher-dimensional state spaces, we will not pursue the splines approach here.

2.2.2.3 Imposing Balanced Error and Collocation Constraints

In this paper, we present an approach to optimal node placement based on balancing the maximum approximation errors of the segments, which is partially motivated by the optimality criterion of Lawson (1964). However, instead of identifying the coefficients by explicit segment-wise error minimization, we impose the collocation constraints (CO) at the breakpoints, which is — together with continuity — sufficient to identify the parameters of a piecewise linear approximation; if higher-order piecewise polynomials are fitted, additional collocation nodes have to be inserted in the interior of the segments. Consequently, in order to find a solution to the uniform approximation and collocation problem with explicit node choice, (2.5), we need to find a vector of nodes and coefficients such that the constraints of (2.5) are satisfied, and the errors of the approximation are balanced.

In particular, let us introduce a slack variable for the cell-wise error

$$z_i = \max_{x_i \leq x < x_{i+1}} |R_{\hat{V}}(x; \mathbf{a}, \mathbf{x})| \quad \forall i \in I. \tag{2.9}$$

Then, (approximate) BE with tolerance $\epsilon_{\mathbf{z}}$ can either be imposed by the $n+1$ equations (BE), or — as we found it to be numerically more efficient — by pairwise comparison of all cell-wise errors

$$\frac{|z_i - z_j|}{z_i} \leq \epsilon_{\mathbf{z}} \quad \forall (i,j) \in I \times I,\ i \neq j. \tag{2.10}$$

Note that equation (2.10) can be reformulated as a linear constraint (see below). Thus, imposing BE yields a system of $n+1$ non-linear equality constraints for the slack variables z_i, and $2n(n+1)$ linear inequality constraints for the actual comparisons.

Combining the BE constraints for optimal node placement, (2.9) and (2.10), with the constraints of (2.5), the following system of equations in the variables $(\mathbf{a}, \mathbf{x}, \mathbf{z})$ identifies a solution to (2.5):

$$R_{\hat{V}}(x_i; \mathbf{a}, \mathbf{x}) = 0, \quad \forall x_i \in \mathbf{x} \tag{2.11a}$$

$$z_i = \max_{x_i \leq x < x_{i+1}} |R_{\hat{V}}(x; \mathbf{a}, \mathbf{x})| \quad \forall i \in I \tag{2.11b}$$

$$(1 - \epsilon_{\mathbf{z}}) z_i - z_j \leq 0 \quad \forall (i,j) \in I \times I,\ i \neq j \tag{2.11c}$$

$$(-1 - \epsilon_{\mathbf{z}}) z_i + z_j \leq 0 \quad \forall (i,j) \in I \times I,\ i \neq j \tag{2.11d}$$

$$x_i + \epsilon_{\mathbf{x}} \leq x_{i+1} \quad \forall x_i, x_{i+1} \in \mathbf{x} \tag{2.11e}$$

where (2.11c) and (2.11d) are the reformulated linear BE conditions, and (2.11e) is the grid validity constraint, that additionally enforces some minimum distance $\epsilon_{\mathbf{x}}$ between grid nodes.

In summary, this section derived a sufficient optimality criterion of the distribution of breakpoints and collocation nodes, which is integrated with the MPEC algorithm in the next section. We conclude this section on balanced errors and its application of uniform approximation and

collocation problems with a remark: In the context of node choice, the mechanics behind equioscillation as an optimality criterion are actually very intuitive: Suppose the global maximum approximation error lies in the interval $[x_i, x_{i+1}]$. Then, slightly shifting the interpolation node x_i to the right (assuming that f and q are not intersecting in the interior of (x_i, x_{i+1})) will decrease the maximum error, while increasing the error of the cell to the left of x_i, thus the L_∞ norm of the approximation will decrease. Thus, q cannot be optimal until the errors in all cells are balanced.

2.2.3 MPEC Estimation with Flexible Grids

We now turn to the question of how to integrate (optimal) node placement with the MPEC approach of estimation of dynamic models. First, note that if $R_{\hat{V}}$ is the residual of the functional equation determining the value function of interest, the optimization problem (2.5) fully determines our function approximation, with degrees of freedom being the coefficients of the polynomials and the nodes themselves. Thus, we replace the collocation constraints in the original MPEC problem (2.3) with problem (2.5) to obtain our new bi-level optimization problem:

$$\max_{\theta,\mathbf{a},\mathbf{x}} L(\theta; \hat{V}_\theta(\cdot, \mathbf{a}), \{\tilde{x}_t, y_t\}_{t=1}^T) \tag{2.12a}$$

$$\text{s.t. } (\mathbf{a}, \mathbf{x}) = \arg\min_{\mathbf{b},\mathbf{y}} \|R_{\hat{V}_\theta}(y; \mathbf{b}, \mathbf{y})\|_\infty \tag{2.12b}$$

$$\text{s.t. (CO), (2.5b) hold.} \tag{2.12c}$$

However, it is not obvious how to obtain a solution to (2.12). While there is a large literature on how to solve this kind of bi-level optimization problems (see, for example, Colson et al., 2007; Fletcher et al., 2006), they remain "notoriously difficult" to solve for the following reason: If we want to avoid solving the system of constraints in every iteration of the solution process (which would then constitute an NFXP approach with an r-adaptive grid using direct minimization), the lower-level minimization problem (2.12b, 2.12c) has to be replaced by something that can be handled by a constrained optimization solver, which is usually systems of equalities and inequalities. One approach is to replace the lower-level optimization problem by its first-order constraints, which are the KKT conditions in case of inequality constraint lower-level problems. However, this procedure will establish complementarity constraints in the new single level problem, which cause severe problems for most algorithms currently available for constrained optimization (see the literature cited above), and is thus a very active field of research in optimization. Instead, we will use the system of (in-)equalities in the following, that constitutes a set of sufficient optimality conditions for our node placement problem based on balanced errors, (2.11), without introducing complementarity constraints into the estimation problem.

Replacing the lower-level optimization problem in (2.12) by the set of sufficient conditions

(2.11), we finally solve the following optimization problem in order to obtain a maximum likelihood estimate of the parameter vector θ using MPEC:

$$\begin{aligned} \max_{\theta,\mathbf{a},\mathbf{x},\mathbf{z}} \; & L(\theta; \hat{V}_\theta(\cdot, \mathbf{a}, \mathbf{x}), \{\tilde{x}_t, y_t\}_{t=1}^T) \\ \text{s.t. } & (2.11) \text{ holds} \end{aligned} \tag{2.13}$$

The statement of problem (2.13) concludes the general description of the method. In the next section, we turn to numerical examples and applications.

2.3 Numerical Experiments and Applications

In this section, we first present several numerical examples to demonstrate the idea of flexible grids from minimizing the L_∞ norm, and the equivalence of imposing the balanced error (BE) constraints. Second, we apply the adaptive grid method to the estimation of the well known bus engine replacement model of Rust (1987) which is modified to feature a continuous mileage state.

2.3.1 Function Interpolation with Flexible Grids – Numerical Examples

In the following examples, we assume that the function we are approximating is known and can be evaluated directly. Consequently, we can write the residual function (2.2) as

$$R_{\hat{f}}(x; \mathbf{a}, \mathbf{x}) = f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$$

and hence we are facing a standard function approximation problem (see Appendix 2.A for a short introduction to function approximation).

For each function approximation problem in this section, we will compare three cases: First, we consider the case of standard interpolation where the residual function is set equal to zero at all the nodes of a fixed grid; this corresponds to solving the system of equations given by equation (CO), which is linear for standard function interpolation problems. Second, we directly minimize the L_∞ norm of the residual function, but at the same time impose the interpolation property as constraints; the corresponding constrained optimization problem is described by problem (2.5). Last, we impose the BE constraints on the residual function to obtain the optimal grid by solving the non-linear system (2.11). This procedure allows us to (i) compare the results of the flexible grid method to standard interpolation over fixed grids, to (ii) demonstrate the equivalence of direct minimization and balanced error conditions, and (iii) compare the different approaches with regard to accuracy and computation time.

We consider three examples: In **Example 1**, we approximate three different functions using polynomial approximation on Chebyshev grids as well as on optimal flexible grids. In particular, we verify our grid adaption by ensuring that the solutions are compatible with properties

we can derive from theory: First, the BE solution must always be at least as good or better then using a fixed Chebyshev grid; second, there are special cases where the Chebyshev grid actually produces a uniform approximation; for a detailed discussion of these properties, see Appendix 2.A.2. Consequently, the results from Example 1 are of a qualitative nature, as they serve as benchmarks for our verification analysis. For the quantitative aspects, **Example 2** demonstrates the advantages of balanced errors using piecewise linear approximations. We show that optimal grids produce significantly smaller approximation errors compared to uniform grids with the same number of nodes. This example is of particular interest as we are going to use piecewise linear approximations for the MPEC problem in Section 2.3.2. Finally, **Example 3** demonstrates that the results also hold for piecewise quadratic polynomial approximations. Note that all examples give evidence that computing optimal grids from the balanced error property is substantially more efficient than directly minimizing the L_∞ norm.

For all examples in this subsection, we use the following parametrisations: The minimum difference between the flexible nodes it set to $\epsilon_{\mathbf{x}} = 0.01$ (none of the grid validity constraints is binding though). For the error tolerance of the BE constraints we use $\epsilon_{\mathbf{z}} = 0$, which proves to be most stable from a computational point of view; we conjecture that this is because otherwise a continuum of solutions exist. In order to compute the cell-wise maximum error, we use a grid search with 100 uniformly distributed nodes. To initialize the algorithm, we first compute an optimal solution with regard to the L_1 norm, which we then use as an initial guess for the solution in the L_∞ norm; this approach increases both numerical efficiency and stability. All computation times are reported including the initialization phase. We use Gauss–Legendre quadrature with 10 nodes in each interval to compute the corresponding integrals for the L_1 norm (see Judd, 1998, for a detailed description of different quadrature methods). All examples are computed in Matlab using the "fmincon" solver with the "sqp" algorithm and default settings. In those examples where we approximate polynomials, their coefficients are randomly generated and reported along with the results.

2.3.1.1 Example 1 – Polynomial Function Approximation

In Example 1, we approximate degree 5 and 6 ordinary polynomials and the function $f(x) = \exp(x^2)$ by a degree 4 Chebyshev polynomial. The purpose of this example is to show that the method produces results that are in line with results from interpolation theory: As stated in Section 2.2.2.1 and Appendix 2.A.2, using Chebyshev nodes for polynomial approximation minimizes the tightest known error bound that can be optimized over the nodes *independently* of f. This is a very strong result as it holds for any continuous function. However, Chebyshev nodes only put an upper bound on the L_∞ norm of the residual, but they are not generally optimal. Hence, we demonstrate that the BE grid yields approximations that are at least as good, but in many cases better than the polynomial interpolation using Chebyshev nodes. As a special case, we present one example for which we know the Chebyshev nodes to be optimal; this example serves as an important benchmark, as it provides a non-trivial closed

form solution for our method to replicate.

Figures 2.2 and 2.3 show the results for the 5 and degree 6 polynomial approximations, respectively, and Figure 2.4 shows the results for $f(x) = \exp(x^2)$; Table 2.1 lists the corresponding error measures and computation times. First, note that for all three functions, imposing the BE constraints and directly minimizing the L_∞ norm yields the same solution. However, comparing computation times and the number of iterations, we find that the BE solution converges significantly faster. We conjecture that this is because by imposing the BE constraints, the solver accommodates for the approximation error in each individual cell of the grid, and thus has more detailed information about how the approximation error over the whole domain is composed, and how it is affected by a potential node movement. On the other hand, this information is mostly lost when minimizing the aggregate value (the maximum over all cell-wise errors) as in the direct minimization. This observation will be confirmed in the following examples and suggests that imposing the BE constraints might be a fast and efficient approach to obtain optimal grids.

For the degree 5 polynomial (Figure 2.3), we find that the flexible grid solutions exactly replicate the Chebyshev nodes (the red optimized nodes coincide with the turquoise Chebyshev nodes). In Appendix 2.A.2, we argue why Chebyshev nodes are optimal in the L_∞ norm for any degree n polynomial interpolated by an degree $n-1$ polynomial. Consequently, this result implies that the optimal grid solution (obtained either by direct minimization or by imposing the BE constraints) is correct.

In case of the degree 6 polynomial (Figure 2.3), we observe that the nodes of the optimal grid do not coincide with the Chebyshev nodes in this particular example; conversely, the interpolation over the Chebyshev grid does not exhibit the BE property. And indeed, the maximum absolute approximation error is significantly smaller for the optimal grid compared to the Chebyshev grid. As mentioned above, this result in line with the theory, as Chebyshev nodes only put an upper bound on the approximation error.

Similarly, Figure 2.4 confirms the findings for the approximation of $f(x) = \exp(x^2)$.

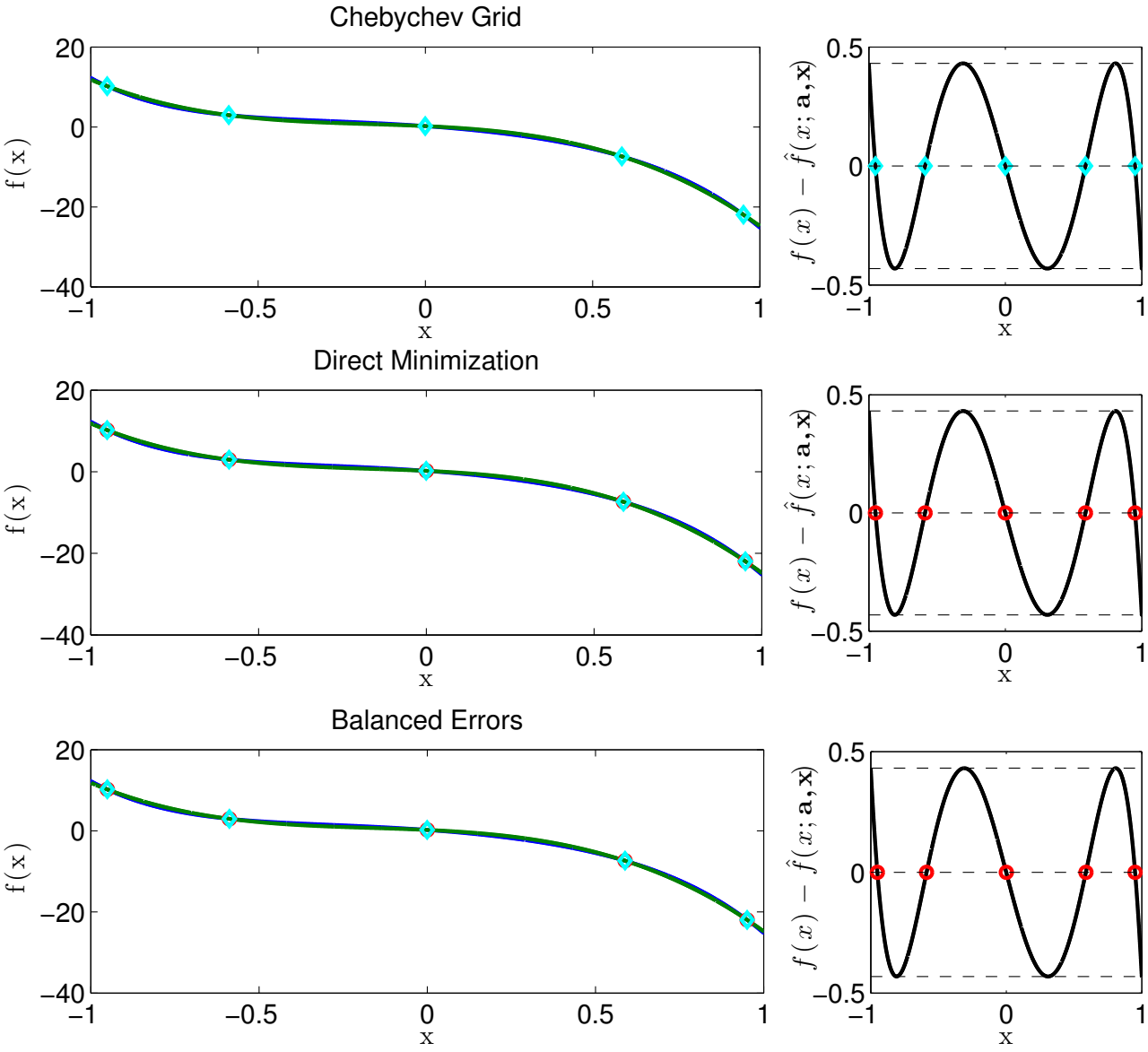
2.3.1.2 Example 2 – Piecewise Linear Function Approximation

In Example 2, we approximate a degree 9 ordinary polynomial by a piecewise linear approximation with 6 nodes, out of which 4 nodes are potentially flexible[8]. In contrast to Example 1, we compare the optimal flexible to a fixed uniform grid approximation, as Chebyshev grids have no interpretation in the context of piecewise approximation.

Figure 2.5 plots the results; the corresponding approximation errors and computation times are stated in Table 2.2. We find that the standard piecewise linear interpolation over a fixed uni-

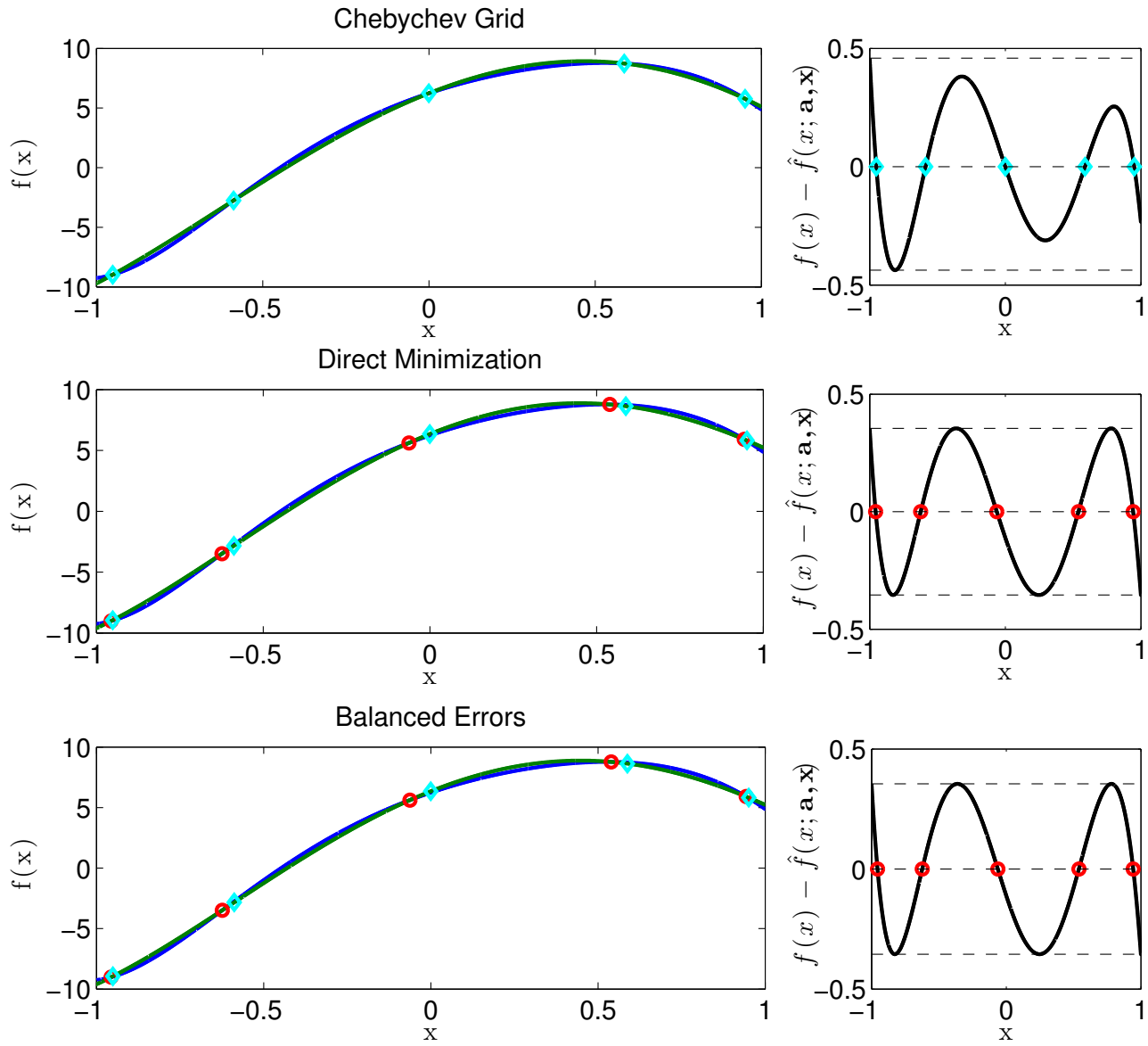
[8]We impose the constraints that $x_1 = x_{min}$ and $x_n = x_{max}$, where x_{min} and x_{max} are the minimum and maximum values of the approximation interval, respectively. This restriction is w.l.o.g. as can be seen in Example 1; in terms of notation used in Section 2.2, it corresponds to $n = 4$.

Figure 2.2: Approximation of a Degree 5 Polynomial Using a Degree 4 Polynomial



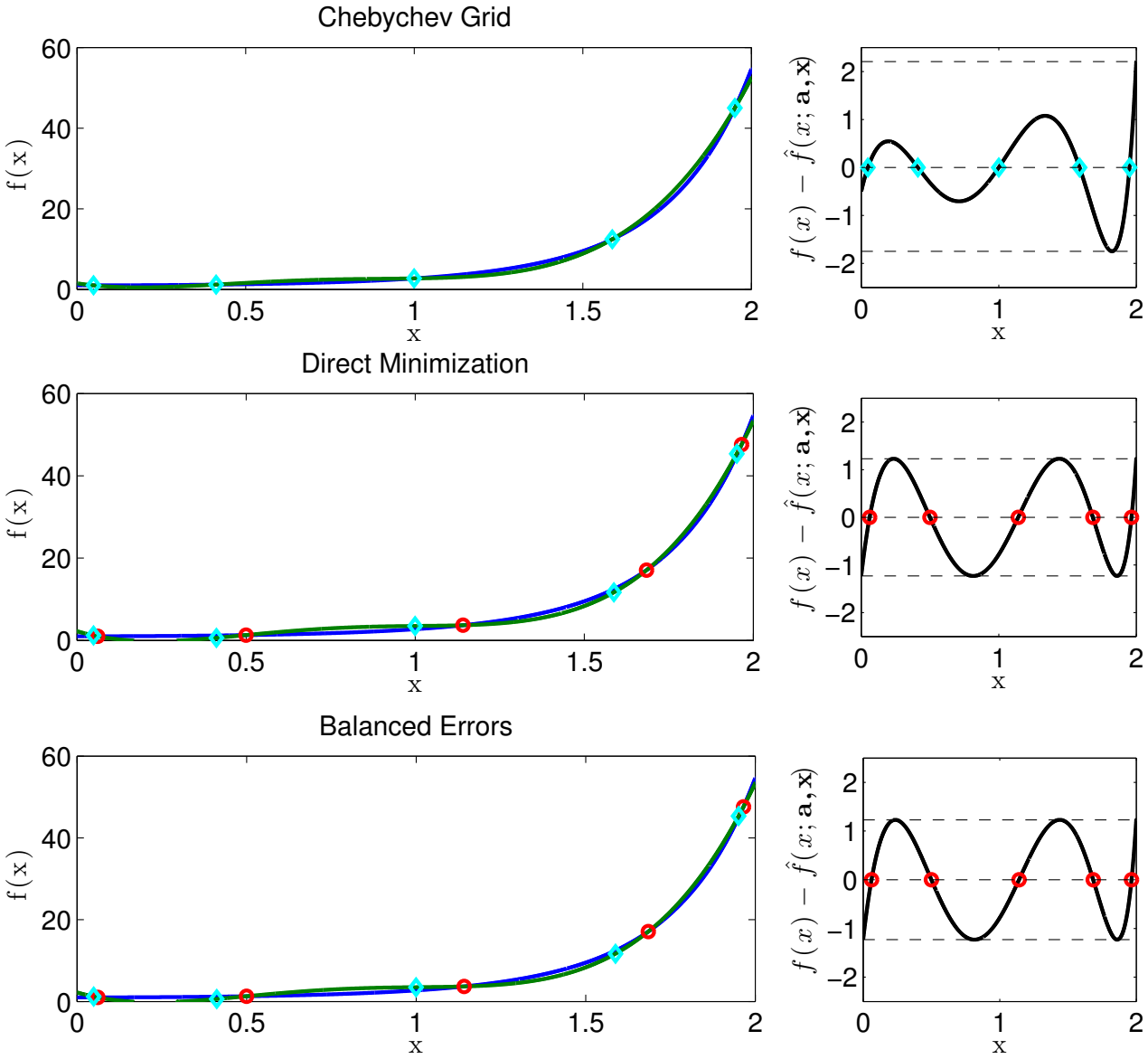
Approximation of a degree 5 ordinary polynomial using a degree 4 Chebyshev polynomial on the interval $[-1, 1]$. The blue line corresponds to the true function, whereas the green line represents the fitted polynomial approximation. Turquoise diamonds and red circles depict the Chebyshev nodes on the interval $[-1, 1]$, and the optimized approximation nodes obtained from imposing the BE conditions or direct minimization, respectively. The plots in the right panel show corresponding residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$. From the top to the bottom, the Figure shows interpolation using Chebyshev nodes, flexible grid with direct minimization of the L_∞ norm, and flexible grid with BE constraints. The coefficients of the true polynomial function f are given by $\boldsymbol{\alpha} = [0.2164, -5.9189, -7.1890, -5.9051, 0.5161, -6.9019]$.

Figure 2.3: Approximation of a Degree 6 Polynomial Using a Degree 4 Polynomial



Approximation of a degree 6 ordinary polynomial using a degree 4 Chebyshev polynomial on the interval $[-1, 1]$. The blue line corresponds to the true function, whereas the green line represents the fitted polynomial approximation. Turquoise diamonds and red circles depict the Chebyshev nodes on the interval $[-1, 1]$, and the optimized approximation nodes obtained from imposing the equioscillation conditions or direct minimization, respectively. The plots in the right panel show corresponding residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$. From the top to the bottom, the Figure shows interpolation using Chebyshev nodes, flexible grid with direct minimization of the L_∞ norm, and flexible grid with BE constraints. The coefficients of the true polynomial function f are given by $\boldsymbol{\alpha} = [6.2356, 9.2929, -9.2861, 3.3064, -0.9446, -5.5323, 1.8073]$.

Figure 2.4: Approximation of $f(x) = \exp(x^2)$ Using a Degree 4 Polynomial



Approximation of $f(x) = \exp(x^2)$ using a degree 4 Chebyshev polynomial on the interval $[0, 2]$. The blue line corresponds to the true function, whereas the green line represents the fitted polynomial approximation. Turquoise diamonds and red circles depict the Chebyshev nodes on the interval $[-1, 1]$, and the optimized approximation nodes obtained from imposing the equioscillation conditions or direct minimization, respectively. The plots in the right panel show corresponding residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$. From the top to the bottom, the Figure shows interpolation using Chebyshev nodes, flexible grid with direct minimization of the L_∞ norm, and flexible grid with BE constraints.

Table 2.1: Comparison of Approximation Errors – Polynomial Approximation

Approximating a degree 5 polynomial with a degree 4 polynomial			
	Chebyshev Grid	Direct Min.	Balanced Errors
L_∞	0.4314	0.4314	0.4314
L_1	0.5457	0.5457	0.5457
Time in Sec.	-	3.77	0.35
# Iterations	-	136	10
Approximating a degree 6 polynomial with a degree 4 polynomial			
	Chebyshev Grid	Direct Min.	Balanced Errors
L_∞	0.4587	0.3548	0.3548
L_1	0.4374	0.4487	0.4487
Time in Sec.	-	2.31	0.38
# Iterations	-	83	12
Approximating $\exp(x^2)$ **with a degree 4 polynomial**			
	Chebyshev Grid	Direct Min.	Balanced Errors
L_∞	2.2058	1.2303	1.2303
L_1	1.2534	1.5566	1.5566
Time in Sec.	-	33.37	0.40
# Iterations	-	1143	13

Approximation errors and computation times of the approximation of a degree 5 polynomial, a degree 6 polynomial, and $\exp(x^2)$ by a degree 4 polynomial, using interpolation over a Chebyshev grid, a flexible grid obtained from direct minimization of the L_∞ norm ("Direct Min."), and a flexible grid obtained from the BE constraints ("Balanced Errors"); the examples correspond to Figures 2.2-2.4

form grid produces large approximation errors, especially in the areas where the approximated function has high curvature, as it is the case between the first and second node for example. Consequently, the flexible grid solutions demonstrate how the approximation error can be reduced by shifting the nodes towards this areas. Looking at the residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$ we find that for both, the direct minimization and the BE solution the errors are balanced, while the error for the uniform grid interpolation strongly varies among the intervals. Note that this example demonstrates the difference between balanced errors, alternating errors, and equioscillation: While the errors of the optimal solutions are obviously balanced — the maximum absolute error is the same for each interval $[x_i, x_{i+1}]$ — they are not alternating (and hence not equioscillating) as the residual does not change sign at all its zeros (including the breakpoints).

Table 2.2 further confirms these findings: The L_∞ norm decreases by an order of magnitude from 11.8250 to 1.5649 for the optimal flexible grid compared to the uniform grid approxi-

mation. Again, we find that imposing the BE constraints yields the same solution as direct minimization, but computation times and the number of iterations are significantly lower for the BE approach.

Table 2.2: Comparison of Approximation Errors – Piecewise Linear Approximation

	Uniform Grid	Direct Min.	Balanced Errors
L_∞	11.8250	1.5649	1.5649
L_1	3.8520	2.1051	2.1051
Time in Sec.	-	1.56	0.58
# Iterations	-	76	18

Approximation errors and computation times of the approximation of a degree 9 polynomial by piecewise linear interpolation, over a fixed uniform grid, a flexible grid obtained from direct minimization of the L_∞ norm ("Direct Min."), and a flexible grid obtained from the BE constraints ("Balanced Errors"); the example corresponds to Figure 2.5.

2.3.1.3 Example 3 – Higher Order Piecewise Polynomial Approximation

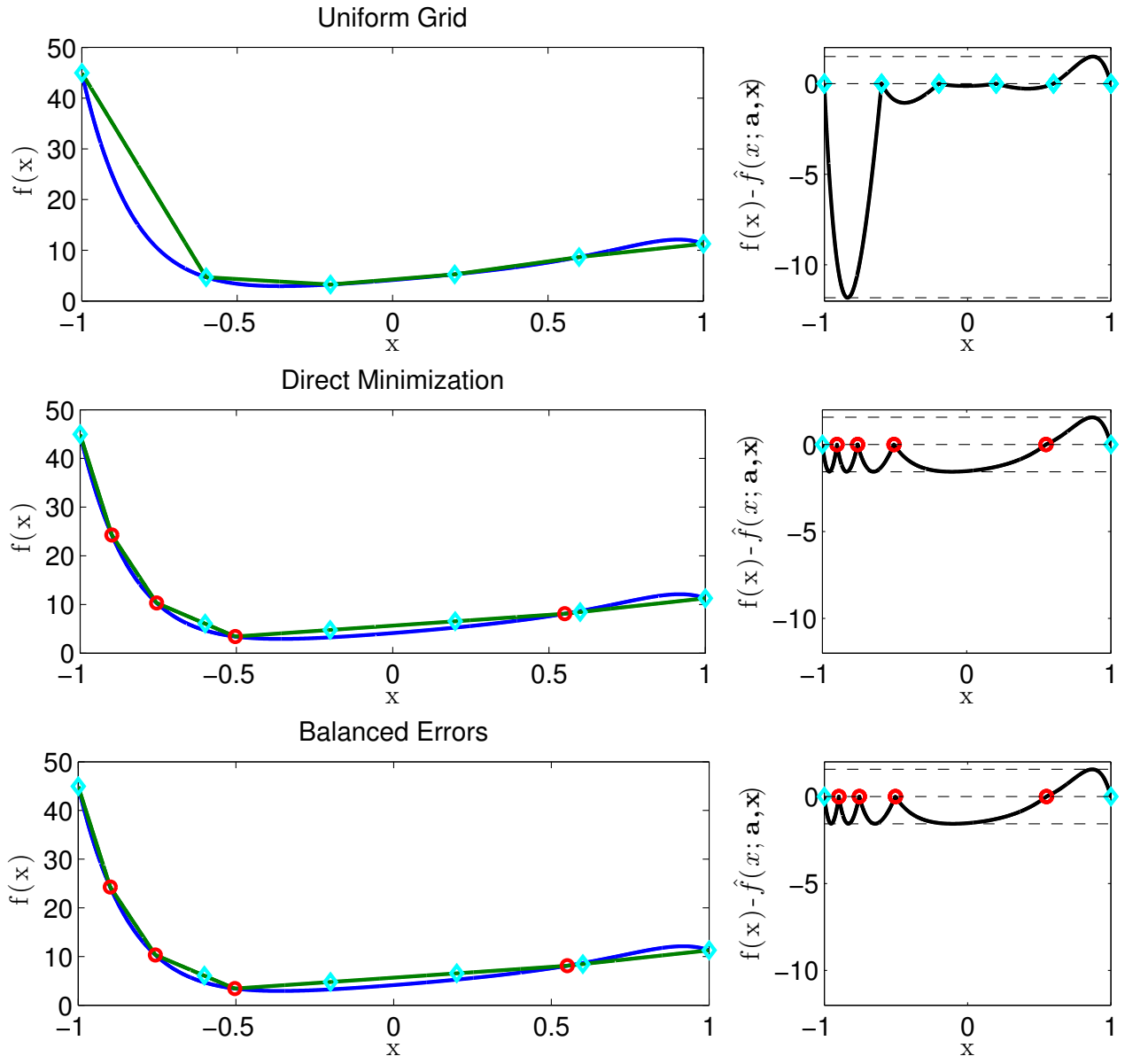
In Example 3, we show that the previously obtained results also hold for higher order piecewise polynomial approximations. For this purpose, we approximate the same function as in Example 2, but by a piecewise quadratic polynomial approximation with 4 nodes out of which 2 are flexible. Note that for simplicity, we distribute the additional interpolation nodes necessary to identify all degrees of freedom uniformly between the breakpoints (see Appendix 2.A.3 for details).

Figure 2.6 plots the corresponding results; the corresponding approximation errors and computation times are stated in Table 2.3. We find that in this example, the uniform grid piecewise polynomial approximation shows large approximation errors between the first two nodes. By allocating the nodes more efficiently, the approximation errors decrease significantly. In particular the maximum absolute error decreases from 5.3260 for the standard interpolation to 1.2731 for the flexible grid. Again, we find that imposing the BE constraints yields the same solution as direct minimization, but computation times and the number of iterations are significantly lower for the BE approach.

2.3.2 MPEC Estimation with Flexible Grids – Numerical Results

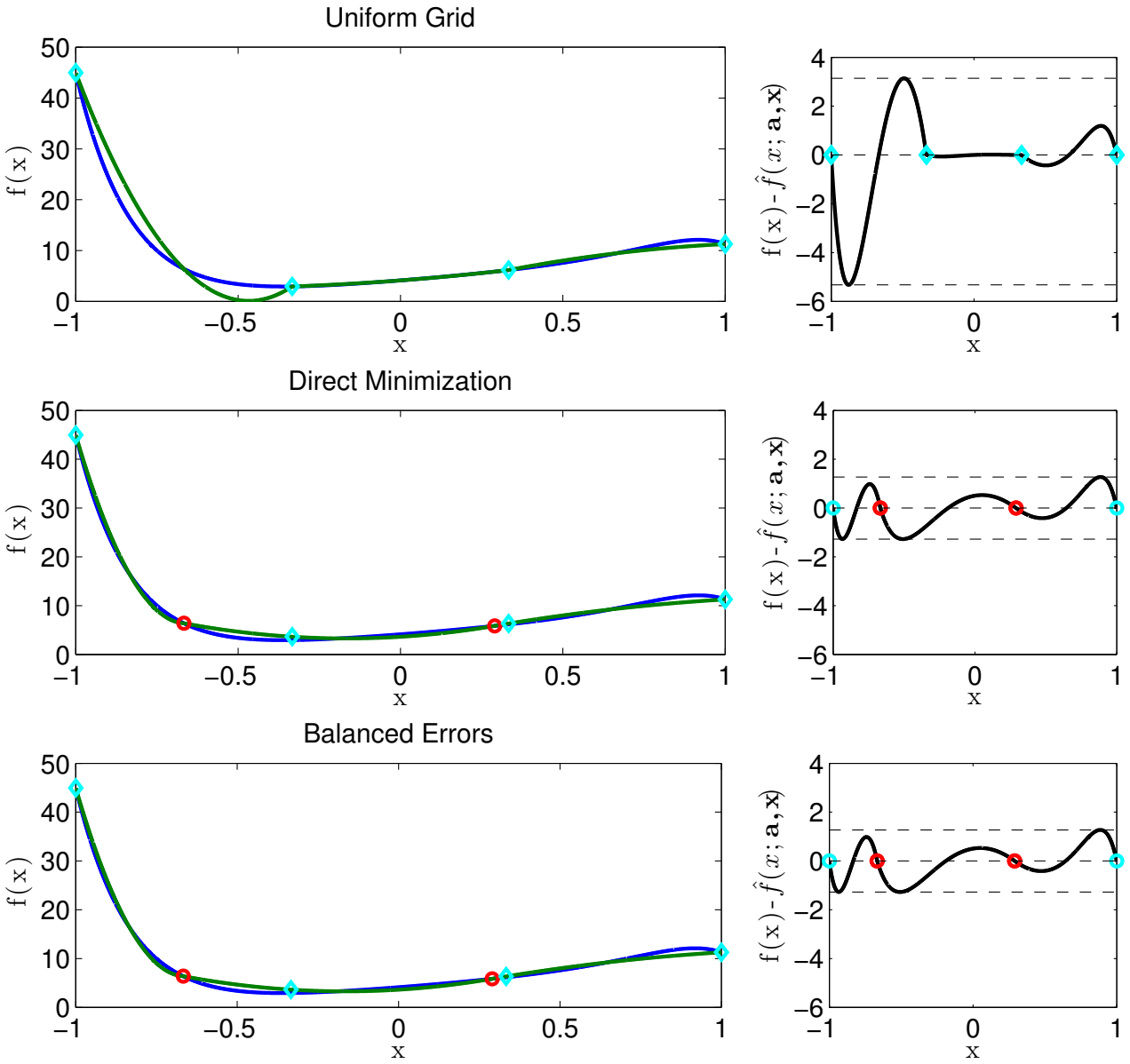
In this section, we apply our balanced error grid adaption method to the estimation of the well known bus engine replacement model of Rust (1987). We begin with a brief description of the model, and then compare the fixed and flexible grid approaches in two steps: First, we present results for fixed model parameters in order to demonstrate the potential advantages of flexible

Figure 2.5: Approximation of a Degree 9 Polynomial Using Piecewise Linear Approximation



Approximation of a degree 9 ordinary polynomial on the interval $[-1, 1]$ by a piecewise linear approximation with 6 nodes, out of which 4 nodes are potentially flexible. The blue line corresponds to the true function, whereas the green line represents the fitted piecewise linear approximation. Turquoise diamonds and red circles depict the fixed uniform nodes, and the optimized approximation nodes obtained from imposing the BE conditions or direct minimization, respectively. The plots in the right panel show corresponding residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$. From the top to the bottom, the Figure shows piecewise linear interpolation using uniformly distributed nodes, flexible nodes with direct minimization of the L_∞ norm, and flexible nodes with BE constraints. The coefficients of the true polynomial function f are given by $\boldsymbol{\alpha} = [4.1239, 2.7956, 5.0862, -1.2933, 7.8788, -7.8582, 9.9192, -2.8339, 3.4032, -9.9500]$.

Figure 2.6: Approximation of a Degree 9 Polynomial Using Piecewise Quadric Approximation



Approximation of a degree 9 ordinary polynomial on the interval $[-1, 1]$ by a piecewise quadratic polynomial approximation with 4 nodes, out of which 2 nodes are potentially flexible. The blue line corresponds to the true function, whereas the green line represents the fitted piecewise polynomial approximation. Turquoise diamonds and red circles depict the fixed uniform nodes, and the optimized approximation nodes obtained from imposing the BE conditions or direct minimization, respectively. The plots in the right panel show corresponding residuals $f(x) - \hat{f}(x; \mathbf{a}, \mathbf{x})$. From the top to the bottom, the Figure shows piecewise polynomial approximation using uniformly distributed nodes, flexible nodes with direct minimization of the L_∞ norm, and flexible nodes with BE constraints. The coefficients of the true polynomial function f are given by $\boldsymbol{\alpha} = [4.1239, 2.7956, 5.0862, -1.2933, 7.8788, -7.8582, 9.9192, -2.8339, 3.4032, -9.9500]$.

Table 2.3: Comparison of Approximation Errors – Piecewise Quadratic Approximation

	Uniform Grid	Direct Min.	Balanced Errors
L_∞	5.3260	1.2731	1.2731
L_1	2.1636	1.1560	1.1560
Time in Sec.	-	1.08	0.66
# Iterations	-	151	14

Approximation errors and computation times of the approximation of a degree 9 polynomial by a piecewise quadratic polynomial approximation, over a fixed uniform grid, a flexible grid obtained from direct minimization of the L_∞ norm ("Direct Min."), and a flexible grid obtained from the BE constraints ("Balanced Errors"); the example corresponds to Figure 2.6.

grids for solving dynamic models; this is similar to the interpolation examples in Section 2.3.1, except that the function to be approximated can not be evaluated directly. Second, we solve the complete maximum likelihood estimation problem using MPEC; therefore, we set up a Monte Carlo experiment with 100 artificial datasets to study and compare the results of the fixed and flexible grid solutions with regard to accuracy and efficiency.

2.3.2.1 The Bus Engine Replacement Model of Rust (1987)

In the bus engine replacement model of Rust (1987), a manager of a fleet of buses repeatedly decides whether or not to replace the engine for each of the buses. His decision is based on the observation of the current mileage state, and some choice and bus specific utility shock. His per period and utility function for one single bus is given by

$$u_\theta(i, x_t) + \varepsilon_t(i)\,, \quad u_\theta(i, x_t) = \begin{cases} -RC & \text{if } i = 1 \\ -c(x_t, \theta_1) & \text{if } i = 0. \end{cases} \tag{2.14}$$

Hence, the manager faces the decision trade-off of replacing the engine at a high fix cost of RC (decision $i = 1$), or just paying the maintenance costs $c(x_t, \theta_1)$ (decision $i = 0$), which increase with the mileage state x_t and depend on the maintenance cost parameter θ_1. $\varepsilon_t(i)$ is the choice specific shock to utility that is observed by the manager, but not by the econometrician. Assuming that the manager behaves dynamically optimal, his value function is given by

$$V_\theta(x_t, \varepsilon_t) = \max_{i \in \{0,1\}} \{u(i, x_t, \theta_1) + \varepsilon_t(i) + \beta E[V_\theta(x_{t+1}, \varepsilon_{t+1})|i, x_t, \varepsilon_t]\} \tag{2.15}$$

where β is the time discount factor and the subscript θ denotes the dependence of the value function on the parameters RC and θ_1. The conditional expected continuation value in equation (2.15) is given by

$$\begin{aligned} EV_\theta(i, x_t, \varepsilon_t) &\equiv E[V_\theta(x_{t+1}, \varepsilon_{t+1})|i, x_t, \varepsilon_t] \\ &= \int V_\theta(x_{t+1}, \varepsilon_{t+1}) Pr(x_{t+1}, \varepsilon_{t+1}|i, x_t, \varepsilon_t, \theta)\ \mathrm{d}(x_{t+1}, \varepsilon_{t+1}). \end{aligned} \tag{2.16}$$

Even though the observed mileage state x_t has a continuous support in reality, it is a common approach to discretize the state space in a finite number of "bins" (see, for example, Su and Judd, 2012; Rust, 1987). In contrast, we do not make this assumption as our solution method is designed for continuous state spaces. Consequently, while the law of motion for discrete Markov states is a matrix, we need a probability density function for the continuous model. As proposed in Rust (1987) we use the exponential function, so $\Delta_{x_{t+1}} = x_{t+1} - x_t$ is exponentially distributed with the rate parameter θ_2. We follow Rust (1987) by assuming that (i) the utility shock $\varepsilon_t(i)$ is extreme value type I iid. distributed, $\varepsilon_t(i) \sim EV1$ iid, and (ii) x and ε are conditionally independent. Under this assumption, closed form solutions for the integral over the unobserved state variables exist, and the EV function for the continuous problem is given by

$$\begin{aligned} EV_\theta(i,x) = \int_0^\infty \log\left(\sum_{i'\in\{0,1\}} \exp\left(u_\theta(i',(1-i)x+\Delta_x) + \beta EV_\theta(i',(1-i)x+\Delta_x)\right)\right) \\ \cdot\, \theta_2 \exp(-\theta_2\Delta_x)\, \mathrm{d}\Delta_x \end{aligned} \tag{2.17}$$

where i' denotes the decision in the next period. We approximate the integral over the observed state by Gauss–Laguerre quadrature, which is a natural choice as it is optimized for the integration over exponential kernels (see Judd, 1998). Finally the log-likelihood function for the full sample of M buses reads

$$\begin{aligned} L(\theta; EV_\theta(\cdot), \{x_t^j, i_t^j\}_{t=1,j=1}^{T,M}) = \sum_{j=1}^{M}\sum_{t=1}^{T} \log\left(\frac{\exp\left(u_\theta(i_t^j, x_t^j) + \beta EV_\theta(i_t^j, x_t^j)\right)}{\sum_{i\in\{0,1\}} \exp\left(u_\theta(i, x_t^j) + \beta EV_\theta(i, x_t^j)\right)}\right) \\ + \sum_{j=1}^{M}\sum_{t=1}^{T} \log\theta_2 \exp(-\theta_2(\Delta_{x_t^j})). \end{aligned} \tag{2.18}$$

2.3.2.2 Approximating the EV Function for Fixed θ

In this subsection, we assume that the model parameters θ are fixed; consequently, we only have to solve the dynamic problem (2.17). This allows us to compare the solutions for a fixed uniform grid and the flexible grid with BE constraints in a simple and demonstrative context.

In particular, we use the following parametrisations of the model and the algorithm: For the model parameter vector θ, we use the original estimates from Rust (1987), given by $RC = 11.7257$, $\theta_1 = 2.4569$, $\beta = 0.99$, and assume $\theta_2 = 1.5$. For the utility function, we use the standard linear costs given by

$$c(x_t, \theta_1) = 10^{-3} \cdot \theta_1 x_t. \tag{2.19}$$

Additionally, we also consider a cubic cost function to introduce more non-linearities into the problem, and hence make the approximation problem of the EV function more interesting:

$$c(x_t, \theta_1) = 10^{-5} \cdot \theta_1 x_t^3. \tag{2.20}$$

For the mileage state, we assume that the maximum mileage is given by $x_{max} = 400$ (similar to Rust, 1987). For the EV function, we use a piecewise linear approximation. We use the same algorithm parametrisation as in Section 2.3.1.

In our analysis, we consider four different approximations for each of the cost functions: a benchmark case using 400 uniformly distributed nodes, the BE solution with 5 nodes out of which 3 are flexible, a uniform fixed grid with as many nodes as the flexible grid, and a uniform grid where the number of nodes is chosen such that the two grids have roughly the same accuracy in terms of the L_∞ norm.

Table 2.4 lists approximation errors, computation times and iteration counts for all approximations. The upper panel shows the results for the linear cost function (2.19). We find that for the benchmark case with 400 nodes, the approximation errors are small with a L_∞ error of $1.7e-4$. The BE solution with 5 nodes has an error of 0.0441, while the error with a uniform grid with equally many nodes is more than twice as large. To obtain the same accuracy with the uniform grid, 10 nodes are needed in this example. Comparing computation times, we find that the uniform grid solution with 10 nodes is still significantly faster compared to the BE grid solution. Hence, in this example it appears to be more efficient to simply increase the number of nodes of the uniform grid instead of using the flexible grid method.

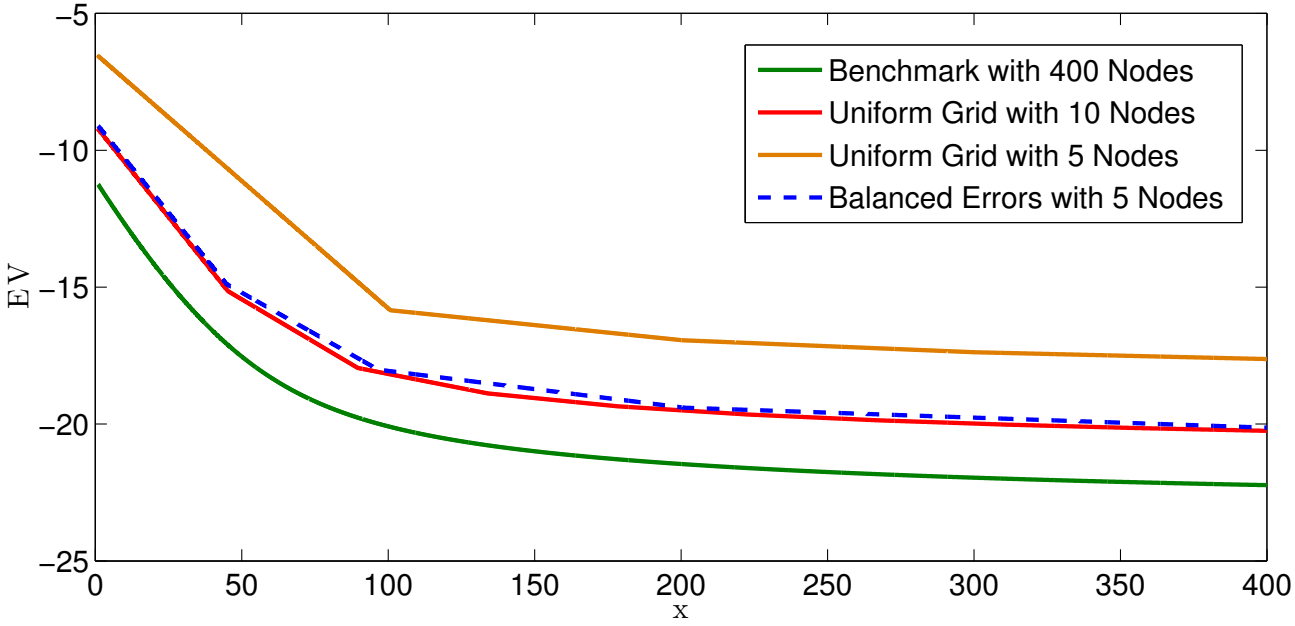
The results turn in favour of the BE solutions when computing the model with the cubic cost function (2.20): In this case, 40 uniformly distributed nodes are needed to obtain the same accuracy as with the BE grid with 5 nodes. Comparing computation times, we find that it actually takes longer to compute the fixed grid solution (0.5201 seconds) compared to the flexible grid (0.4391 seconds). Figures 2.7 and 2.8 depict the four different approximations of the EV functions for the linear and the cubic cost functions, respectively, and illustrate why this is the case: Using the cubic cost function, we find that most curvature is massed at the low mileage states, while it is almost linear or even constant otherwise. This makes the approximation of the EV function by piecewise linear segments over a uniform grid very inefficient, as the nodes should be placed in regions of high curvature. Conversely, the BE grid efficiently moves the nodes to the critical part of high curvature, and therefore achieves much higher accuracy using an equal amount of nodes. Hence, even in such a simple example, and in particular isolated from the full estimation problem, the flexible grid solution can be more efficient compared to approximation over uniform grids.

So far, we have assumed that the parameter vector θ is fixed, and thus only the dynamic model needs to be solved. The next subsection addresses the complete MPEC problem using simulated data.

2.3.2.3 Monte Carlo Study for the MPEC Problem

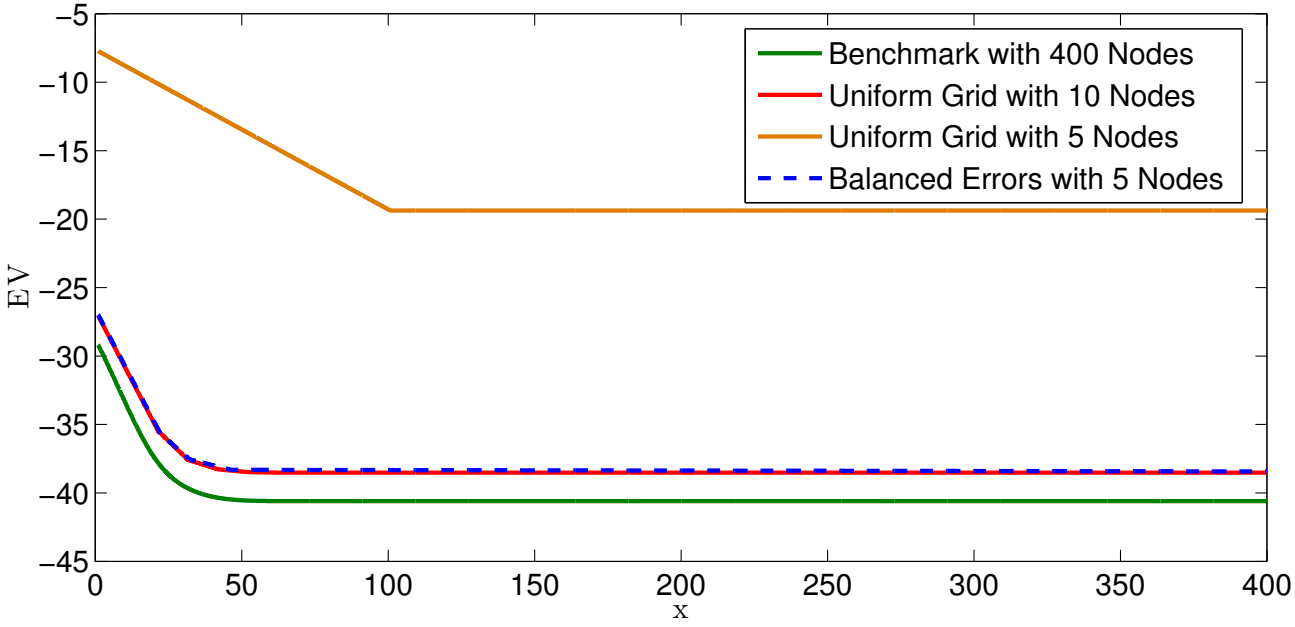
In this subsection, we estimate the parameters of the bus engine replacement model of Rust (1987), using the flexible grid approach with MPEC, as described by problem (2.13), and

Figure 2.7: EV as a Function of x for the Linear Cost Function (2.19)



EV as a function of mileage state x for fixed θ and linear cost function (2.19).

Figure 2.8: EV as a Function x for the Cubic Cost Function (2.20)



EV as a function of mileage state x for fixed θ and cubic cost function (2.20).

Table 2.4: Approximation of the *EV* Function of the Rust (1987) Model

	Linear Cost Function			
	Benchmark	Uniform Grid 1	Uniform Grid 2	Balanced Errors
L_∞	0.0002	0.1054	0.0436	0.0441
L_1	0.0100	10.0561	3.1749	11.2941
Time in Sec.	8.66	0.08	0.16	0.43
# Iterations	8	8	7	13
# Nodes	400	5	10	5
	Cubic Cost Function			
	Benchmark	Uniform Grid 1	Uniform Grid 2	Balanced Errors
L_∞	0.0022	4.3390	0.1151	0.1120
L_1	0.0324	173.0533	2.0781	22.9348
Time in Sec.	12.02	0.06	0.52	0.44
# Iterations	6	4	6	17
# Nodes	400	5	40	5

Approximation errors, computation times and iteration counts of the approximation of the *EV* function (2.17) for a fixed parameter vector θ. The upper and lower panel list the results for the linear cost function (2.19) and the lower panel for the cubic cost function (2.20), respectively. Besides the benchmark solution with a uniform grid of 400 nodes, the tables lists the BE solution with 5 nodes out of which 3 are flexible ("Balanced Errors"), a uniform grid solution with 5 nodes ("Uniform Grid 1"), and a uniform grid solution where the number of nodes is chosen to roughly match the L_∞ norm of the BE solution ("Uniform Grid 2").

compare it to the standard fixed grid solution by solving problem (2.3).

In order to obtain a measure of variation, we simulate 100 datasets with 500 buses each running for 150 periods. For the simulation of the data we use the same parameters as in the previous example given by $\beta = 0.99$, $RC = 11.7257$, $\theta_1 = 2.4569$ and $\theta_2 = 1.5$, furthermore, we use a linear cost function as in (2.19). For each dataset, we use three starting points for θ, given by $(RC, \theta_1) \in \{(2,1),(10,3),(17,5)\}$ (one significantly smaller, one close to, and one significantly larger than the true parameters). Also, as in the original model, the parameter of the mileage state transition, $\theta_2 = 1.5$, can be estimated independently, which is why we focus on the estimation of the cost parameters RC and θ_1.[9] To compute the one period ahead value expectations in equation (2.17), we use 10 node Gauss–Laguerre quadrature rules.

As in the previous section, we use four different approximations of the expected value function: a benchmark solution using 400 of uniformly distributed nodes, the BE solution with 5 nodes out of which 3 are flexible, a uniform fixed grid with as many nodes as the flexible grid, and

[9]To avoid the multicollinearity issue pointed out by Rust (1987), we also assume β to be fixed.

a uniform grid where the number of nodes is chosen to roughly match the relative root mean squared error ($rRMSE$)[10] of the BE grid.

Table 2.5 lists parameter estimates, errors with respect to the benchmark solution (rRMSE), computation times and iteration counts for all approximations. We find that for the 5 node uniform grid, the estimates dramatically differ from the benchmark solution, and are not even within 4 standard deviations; the error of the 5 node uniform grid is very large given an $rRMSE$ of 0.3161. The BE solution with 5 nodes out of which 3 are flexible produces much more accurate results, with an $rRMSE$ of 0.0616. To obtain the same level of accuracy, a uniform grid with 17 nodes is needed. Comparing efficiency, we find that the BE solution is about 1.5 times faster than the uniform solution with the same accuracy. Also, we find that for all approximation methods, all runs converged, which indicates that also the BE method is sufficiently stable.[11]

We conclude the discussion of the results by noting that by using flexible grids with BE constraints in the context of MPEC estimation of dynamic models, one can potentially harvest significant gains in efficiency, compared to standard uniform grid approximations.

2.4 Conclusion

2.4.1 Summary

In this paper we show how to integrate flexible interpolation grids with the estimation of dynamic models using the MPEC approach of Su and Judd (2012). We derive a set of conditions to enforce balanced errors (BE), which we argue to be sufficient for optimality. In particular we make use of the equioscillation theorem to obtain value function approximations that are optimal in the L_∞ norm, given their functional form and the total number of grid nodes.

We demonstrate the equivalence of minimizing the L_∞ norm directly using non-linear optimization and imposing BE constraints in several numerical experiments. We observe that in all cases considered, computations using the BE constraints are significantly faster than direct minimization. This finding suggests that our approach integrated with MPEC might be a fast and efficient way to obtain precise parameter estimates using optimal grids.

[10] We define the $rRMSE$ as

$$rRMSE = \sqrt{\frac{1}{J}\frac{1}{K}\sum_{j=1}^{J}\sum_{k=1}^{K}\left(\frac{\hat{RC}^{j,k} - \hat{RC}_B^{j,k}}{\hat{RC}_B^{j,k}}\right)^2 + \left(\frac{\hat{\theta}_1^{j,k} - \hat{\theta}_{1,B}^{j,k}}{\hat{\theta}_{1,B}^{j,k}}\right)^2} \tag{2.21}$$

where J is the number of datasets, K is the number of initial guesses per dataset, $\hat{RC}^{j,k}$ and $\hat{\theta}_1^{j,k}$ are the parameter estimates for dataset j and initial guess k, and the subscript B denotes the benchmark solution.

[11] Note that we always initialize the flexible grid method from a feasible starting point. Therefore, we first compute the uniform grid solution with equally many nodes to obtain estimates $\tilde{\theta}$. Second, we solve the BE constraints problem to obtain a feasible grid for $\tilde{\theta}$. All computation times are reported including this initialization.

Table 2.5: Results for the MPEC Estimation of the Rust (1987) Model

	Benchmark Solution	Uniform Grid 1	Uniform Grid 2	Balanced Errors
RC	11.7697	8.6682	11.3967	11.0834
	(0.3956)	(0.1369)	(0.3493)	(0.3098)
θ_1	2.4886	2.0585	2.3435	2.4558
	(0.1405)	(0.1223)	(0.1248)	(0.1255)
$rRMSE$	-	0.3161	0.0661	0.0616
Time in Sec.	365.84	2.90	12.01	7.59
# Nodes	400	5	17	5
# Conv. Runs	300	300	300	300

Mean and standard deviation estimates of the parameters RC and θ_1 from 300 Monte Carlo run. The root mean square error ($rRMSE$) is reported as a measure of variation of the estimates from the Benchmark solution. Also reported are computation times, the number of grid nodes, and the number of runs converged. Besides the benchmark solution with a uniform grid of 400 nodes, the tables lists the BE solution with 5 nodes out of which 3 are flexible ("Balanced Errors"), a uniform grid solution with 5 nodes ("Uniform Grid 1"), and a uniform grid solution where the number of nodes is chosen to roughly match the $rRMSE$ of the BE solution ("Uniform Grid 2").

We apply our method to the well known bus engine replacement model of Rust (1987) — modified to feature a continuous mileage state — and compare our results to standard uniform grid approximations with regard to accuracy and efficiency: First, using fixed model parameters we find that the BE grid can significantly reduce approximation errors compared to a uniform grid with equally many nodes. Furthermore, we show that if the approximated function is of sufficient complexity, our solution method has also better efficiency; conversely, a fixed grid solution that achieves the same level of accuracy as the flexible grid takes considerably longer computation time.

Second, we compute solutions for the full maximum likelihood estimation problem of the bus engine replacement model on simulated data using MPEC with flexible grids. We find that the parameter estimates of the BE grid approach are significantly closer to the true parameter estimates compared to the fixed grid solution with equally many approximation nodes. Moreover, a uniform grid solution where the number of nodes is chosen to match the accuracy of the flexible grid takes considerably longer computation time. Consequently, we conclude that using the BE grid approach developed in this paper to estimate dynamic models using MPEC with flexible grids can lead to significant gains in efficiency and accuracy compared to the commonly used fixed grid approaches.

2.4.2 Outlook

The current implementation of our method is limited to models with one continuous state variable. Also, to our knowledge, the optimality conditions for function approximation based on equioscillation or BE are derived for one-dimensional functions only. However, we conjecture that conceptually, the approach carries well over to multiple dimensions. Moreover, we expect the comparative advantage of node movement over fixed grids to increase with dimensionality, as the number of nodes in tensor product grids increases exponentially. Consequently, saving grid nodes by moving them as to balance errors could be even more attractive if dimensionality grows larger.

Therefore, future research will include the experimental evaluation of the multi-dimensional generalization of optimal grids and BE, and, if successful, identify or derive the necessary theory. Additionally, we have to evaluate and implement appropriate finite element geometries and approximation schemes, for which (i) BE can be identified, and (ii) that feature the necessary continuity (and possibly smoothness) properties over the kind of unstructured grid we employ. This will finally allow us to apply our method to applications that feature the dimensionality of modern dynamic models.

Besides dimensionality, we will generalize our method to feature locally approximation error minimizing polynomial segments in the sense of Lawson (1964). Conceptually, this can be implemented by applying the equioscillation theorem for each segment, and balance the resulting errors across segments. Such a procedure would put more rigour on our method as the resulting interpolant is provably the best piecewise polynomial approximation of degree k with n breakpoints.

2.A Function Approximation

Suppose an unknown function $f : \mathbb{R} \supseteq D \to \mathbb{R}$ is to be represented on a computer. While there are many functions for which a finite-dimensional representation exists, this is generally an infinite-dimensional problem; also, even if such a representation exists, it might be unknown, and thus one might need to approximate the function from a finite number of evaluations. There are several popular approaches to function approximation, out of which we briefly introduce polynomial approximation, piecewise polynomial approximation, and splines, mainly to define the nomenclature used in the paper.

2.A.1 Polynomial Approximation

In many function approximation schemes, the approximating function $\hat{f}(\cdot; \mathbf{a})$ is composed as a weighted sum of basis functions $\Phi^n \equiv \{\varphi_i\}_{i=0}^n$, with weight vector $\mathbf{a} \equiv (a_i)_{i=1}^n$:

$$\hat{f}(x; \mathbf{a}) \equiv \sum_{i=0}^{n} a_i \varphi_i(x) \tag{2.22}$$

The task of approximating f is thus twofold: First, a suitable set of basis function has to be identified, and second, the parameters of the function approximation — in our case the weights on the basis functions — have to be identified, such that the approximation is "as close as possible" to the approximated function.

In polynomial approximation, the basis functions used to form $\hat{f}$ are polynomials of degree n or less, and thus the approximation is itself an element of the space of all polynomials of degree n or less, $\hat{f} \in \mathcal{P}^n$. Consequently, the set of basis functions used to form (2.22) is often chosen to form an orthogonal basis of $\mathcal{P}^n$; popular choices are the Chebyshev, the Hermite, or the Laguerre polynomials (see, for example, Judd, 1998, p. 204). Of course, a naive approach is to set Φ^n equal to the set of all monomials of degree n or less; however, this can lead to serious numerical problems when computing the weights $\mathbf{a}$.

The second problem is to find parameters such that the quality of approximation is "good". The most widely used approaches are based on one of the two concepts, least squares minimization or interpolation.

Define the residual

$$R_{\hat{f}}(x; \mathbf{a}) \equiv f(x) - \hat{f}(x; \mathbf{a}) \tag{2.23}$$

The least squares approach minimizes the weighed squared errors over the domain of approximation

$$\min_{\mathbf{a}} \int_D R_{\hat{f}}(x; \mathbf{a})^2 w(x) dx \tag{2.24}$$

where w is a non-negative weighting function, imposing "priorities" on the domain of approximation. Depending on the algorithm in use, different variants of computing the integral in (2.24), and different weighting functions w can be applied.

On the other hand, interpolation ensures that the approximation equals the function at a specific set of interpolation nodes $\mathbf{x} \equiv (x_i)_{i=1}^n$

$$\hat{f}(x_i; \mathbf{a}) = f(x_i), \ \forall x_i \in \mathbf{x} \tag{2.25}$$

Here, the parameters $\mathbf{a}$ are the solution to the linear system of equations (2.25), which is square if $|\mathbf{a}| = |\mathbf{x}| = n$. While the interpolation approach is easy and intuitive, its result is not as rigorous as the least squares approximation error minimization; a way to combine the two is the Chebyshev regression approach (see Judd, 1998, p. 223).

2.A.2 Chebyshev Nodes

An important special case of interpolation node choice for polynomial approximation are the Chebyshev nodes. Suppose a function f on $[-1, 1]$ is interpolated at n nodes $\mathbf{x} \equiv (x_i)_{i=1}^n$ by $\hat{f} \in \mathcal{P}^{n-1}$ such that (2.25) holds. Then, one can show that the residual as defined by (2.23) is (see, for example, Judd, 1998, Thm. 6.7.1)

$$R_{\hat{f}}(x; \mathbf{a}) = \frac{f^{(n)}(\xi(x))}{n!} \ \Psi(x; \mathbf{x}) \tag{2.26}$$

for some $\xi(x) \in [-1, 1]$, and where $\Psi(x; \mathbf{x}) \equiv \prod_{i=1}^n (x - x_i)$.

Consequently, a natural approach to (static) node choice is

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\Psi(x; \mathbf{x})\|_\infty \tag{2.27}$$

since it is independent of the function f to be approximated. Note that $\Psi(x; \mathbf{x})$ is monic. If we choose the x_i's to be the roots of the degree n Chebyshev polynomial,

$$\tilde{x}_i \equiv \cos\left(\frac{(2i-1)\pi}{2n}\right), \tag{2.28}$$

$\Psi(x; \tilde{\mathbf{x}})$ can shown to be the L_∞ minimizing polynomial among all monic polynomials of degree n, and therefore constitutes a solution to problem (2.27); thus, $\mathbf{x}^* = \tilde{\mathbf{x}}$. Moreover (see, for example, Judd, 1998, Thm. 6.7.2)

$$\|\Psi(x; \tilde{\mathbf{x}})\|_\infty = 2^{1-n}. \tag{2.29}$$

Consequently, using Chebyshev nodes bounds the interpolation error from above by[12]

$$\|R_{\hat{f}}\|_\infty \leq \frac{\|f^{(n)}\|_\infty}{n!} \ 2^{1-n}. \tag{2.30}$$

This result has great practical implications, as the interpolation nodes can be computed independently of f. Furthermore, if f is sufficiently smooth, one can show that the approximation $\hat{f}$ converges as the number of Chebyshev interpolation nodes is increased (see, for example,

[12]In order for bound (2.30) to be well defined, the function f must be n times continuously differentiable; other bounds for less smooth functions exist; see, for example, Judd (1998), Equation 6.7.5.

Judd, 1998, Thm. 6.7.3), which is not necessarily true for general grid choices (such as uniform) in conjunction with polynomial approximation.

It is important to note that Chebyshev nodes do not necessarily minimize the actual interpolation error; rather, they minimize the portion of the error that is independent of the function to be approximated. However, we want to highlight to interesting special cases: First, suppose the f is such that it's nth derivative is constant. Then, the residual (2.26) is constant in ξ, and thus $\mathbf{x}^*$ minimizes the total maximum absolute interpolation error. For example, if $f \in \mathcal{P}^n$ and $\hat{f} \in \mathcal{P}^{n-1}$, thus if we approximate an degree n polynomial by an degree $n-1$ interpolating polynomial, using Chebyshev nodes minimizes the L_∞ norm of the residual and thus results in a uniform approximation. Second, for any function with $f^{(n)} = 0$, the approximation is exact, independent of the node choice; this includes, for example, any $f \in \mathcal{P}^k$, $k \leq n-1$.

2.A.3 Piecewise Polynomial Approximation and Splines

If the approximating polynomial has full support over D, features of the approximated function in one region can have substantial impact on the approximation quality also in other regions. A well known example is the fact that regions of steep gradients can cause polynomial approximations to oscillate more in all parts of the approximated function.

A popular way to address this issue is piecewise polynomial approximation: Instead of a polynomial with full support, the domain of approximation is subdivided by a grid $\mathbf{y} \equiv (y_i)_{i=1}^m$, and lower-degree polynomials with support only over the respective grid cell (and sometimes its neighbours) are fitted, using interpolation for example. Formally, the interpolant is composed as

$$\hat{f}(x; \mathbf{a}, \mathbf{y}) \equiv \sum_{i=0}^{n} a_{ij} \varphi_i(x), \; x \in [y_j, y_{j+1}) \tag{2.31}$$

Note that if the break points of the approximation are used as interpolation nodes, thus if $\mathbf{x} = \mathbf{y}$, the approximation will automatically be continuous, but not smooth in general.

As polynomial approximation, the user is faced with two problems, namely what degree of basis function polynomial to use, and how to identify the degrees of freedoms (the coefficients); however, with piecewise methods, these problems are slightly more interconnected: If only the break points are used as interpolation nodes, only $|\mathbf{y}| = m$ equations exist to identify the coefficients in (2.31). However, this limits the degree of the local polynomials to one, if no additional constraints are imposed. Thus, the approximation is a combination of piecewise linear segments. While this is straightforward to handle and thus not put any additional complications on identifying $\mathbf{a}$, one usually needs a large amount of break and interpolation points even for nice and smooth functions.

Two popular approaches exist to apply higher-order polynomials: First, additional interpolation nodes can be inserted in the interior of the cells. This procedure generates additional equations for the approximation problem, which in turn identify the coefficients of the higher-

order polynomial terms in (2.31). For simplicity, we distribute the additional interpolation nodes uniformly between the breakpoints of the piecewise polynomial, without making this explicit in the collocation equation (CO) to not overload the notation. Thus, the set of breakpoints is a strict subset of the set of interpolation nodes. In the outlook of Section 2.4.2 we argue on how to obtain rigorously optimal piecewise polynomials by relaxing the simplifying assumption that the breakpoints coincide with interpolation nodes, and by making the additional nodes flexible as well.

Second, additional constraints on the derivatives of the approximation can be imposed, since its functional form (and thus its derivatives) is know. Usually, the constraints impose equality of the derivatives at the breakpoints, in order to ensure smooth approximating functions. This form of approximation is called splines. Formally, the parameters of a polynomial spline of order k are obtained by solving the following linear system of equations:

$$f(x_i) = \hat{f}(x_i; \mathbf{a}, \mathbf{y}),\ \forall x_i \in \mathbf{x} \equiv \mathbf{y} \tag{2.32a}$$

$$\hat{f}_i^{(h)}(x_{i+1}; \mathbf{a}, \mathbf{y}) = \hat{f}_{i+1}^{(h)}(x_{i+1}; \mathbf{a}, \mathbf{y}),\ h = 1, \ldots, k-2,\ i = 1, \ldots, n-2 \tag{2.32b}$$

$$\hat{f} = 0\big|_{\partial D} \tag{2.32c}$$

where $\hat{f}_j(x; \mathbf{a}, \mathbf{y}) = \sum_{i=0}^{n} a_{ij}\varphi_i(x)$ is the jth segment of the spline, and (2.32c) is a generic boundary condition necessary to identify all degrees of freedom. Note that

$$\hat{f}(\cdot; \mathbf{a}, \mathbf{y}) \in \mathcal{C}^{k-2} \tag{2.33}$$

thus a spline of order k is $k-2$ times continuously differentiable. A different approach to spline approximations that relates to the idea of composing an approximation from an orthogonal basis are B-splines, which form a basis of the space of all order k splines; for details, see de Boor (2001).

Essay 3

Self-Organization in Agent-Based Market Models

Market Self-Organization under Limited Information

Gregor Reich[1]
Dept. of Business Administration
University of Zurich
Moussonstrasse 15
8044 Zurich, Switzerland
gregor.reich@business.uzh.ch

August 2011

Abstract

The process of gradually finding an economic equilibrium, the so called tâtonnement process, is investigated in this paper. In constrast to classical general equilibrium modelling, where a central institution with perfect information about consumer preferences and production technologies ("Walrasian auctioneer") organizes the economy, we simulate this process with learning consumer and producer agents, but no auctioneer. These agents lack perfect information on consumption preferences and are unable to explicitly optimize utility and profits. Rather, consumers base their consumption decision on past experience – formalized by reinforcement learning – whereas producers do regression learning to estimate aggregate consumer demand for profit maximization. Our results suggest that, even without perfect information or explicit optimization, it is possible for the economy to converge towards the analytically optimal state.

Keywords: Agent-based computational economics; market self-organisation; reinforcement learning; regression learning.

Note: This paper has been published in: Advances in Artificial Intelligence, 7023:32–41. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer.
The original publication is available at www.springerlink.com
`http://link.springer.com/chapter/10.1007/978-3-642-25274-7_4`

[1]I thank Dietmar Maringer for many helpful discussions. In addition, I thank three anonymous reviewers for helpful comments. Financial support of the WWZ Forum under grant D-131 is gratefully acknowledged.

3.1 Introduction

Ever since the introduction of the concept of competitive market equilibrium by Leon Walras Walras (1954), economists have thought about the question how markets could eventually get to this state. And indeed, this is a reasonable question, since economic equilibrium is defined as a set of prices and endowments, such that every market participant is maximizing her profit or utility. Even more so, if one admits that the actually acting entities in the real world are people, possibly highly heterogeneous, that do not always decide based on solutions to complex optimization problems, but rather based on simple heuristics arising from experience and limited foresight.

To find a plausible explanation of how markets could eventually reach equilibrium, economic research focussed on finding different kinds of processes of price and trade quantity adoption. The first one was introduced by Walras himself, and is based on a central authority called the "Walrasian auctioneer", who, by knowing all agents' preferences and technologies, step by step adjusts the prices until all agents find themselves being in an optimal state. This process is obviously not very realistic for the decentralized markets found in reality, and moreover, has been shown to be stable only under very restrictive assumptions. Other processes developed are the Edgeworth, the Hahn and the Fisher processes; but all of them still rely on very strict assumptions on agents' rationality and optimization abilities, and only the Fisher process does not involve centralized planning. See Hsieh and Mangum (1986) for an overview of the processes and the stability issue.

A more recent development is the modelling of economies and its markets with Agent-based modelling techniques. These models have been applied widely in the field; see Tesfatsion and Judd (2006) for a review of the foundations and paradigms behind Agent-based computational economics, and Rouchier (2008) for a recent survey of the different applications. Recently, some contributions to the Agent-based computational economics literature brought equilibrium dynamics of markets and its comparison to theoretical benchmarks into focus Gintis (2007, 2011). Interestingly, they find that completely decentralized markets might organize, such that prices arising from bilateral trade converge towards the theoretically optimal (hence profit and utility maximizing) values under certain conditions. However, they still assume the agents to choose their actions by solving (constrained) optimization problems to maximize their utility, which is known to every single agent in its functional form.

Since we neither believe in peoples' perfect rationality and optimization abilities, nor in the assumption that people know "what makes them most happy" as a function of their actions (the utility function), we follow a different approach: In this paper, we model a consumption market where certain agents produce goods that other agents consume. We assume that everybody wants to make herself as "happy" as possible, namely by *implicitly* optimizing her utility from consumption by choosing combinations of goods that proved successful in the past. This is implemented with reinforcement learning, an algorithm whose application in economics has been pioneered by Erev and Roth (1998) and that has been used to study economic

decision making and market dynamic analysis since than (see e. g. Rieskamp, Busemeyer, and Laine (2003); Weisbuch and Herreiner (2000); Vriend (2001)); also, multi-agent reinforcement learning is used to study learning in stochastic games and its convergence and optimality properties (see e. g. Bowling and Veloso (2002); Wang and Sandholm (2003), and Busoniu, Babuska, and De Schutter (2008) for a more general survey). At the same time, the agents producing consumption goods try to maximize their profits from product sales. The problem arising from imperfect information about consumer preferences is, that profit maximization involves knowing the trade off between rising a price to increase earnings per unit, and a decreasing demand for the good, since consumer buy less because they cannot afford them any more or because they switch to cheaper alternatives. This trade off, as a map between prices and demand quantities, is generally not known in this model quantitatively (how could it be, if not even consumers know their own preferences as a function of goods prices), but has to be estimated instead. We therefore use a regression learning algorithm (see Spall (2003)) to let producers learn the demand curve of their respective goods to maximize profits.

The question of this paper is whether our market model is able to organize itself without being lend a hand by a central authority. Furthermore, if this is the case, we want to check whether this stable state somehow corresponds to an economic equilibrium, where everybody has maximal attainable utility or profit. Therefore, we will parametrize the model such that we are able to calculate the analytical solution to the so called social planner optimization problem, i. e. the solution of which makes everybody ending up optimally. We than simulate our model and compare the results to this benchmark.[2]

3.2 The Model

On this artificial consumption goods markets, consumer and producers are repeatedly interacting in the following way: At the beginning of a trading period t, each producer j (a total of J) produces a fixed quantity of one unique consumption good and announces its price $p_{t,j}$. Prices and quantities are announced simultaneously for all goods and cannot be changed throughout the period. Once all prices have been announced, consumers (a total of I) repeatedly buy and immediately consume goods as long as (i) the goods are available (and hence not sold out) and (ii) their remaining income is greater or equal to the goods' respective prices. We assume that excess demand is (quantitatively) observable by the producers.[3] Consumers' aim is to

[2] Using the convergence towards some kind of equilibrium state as the only measure for a model's or a learning algorithm's quality has been criticized in both economics Rothschild (2010) and artificial intelligence Shoham, Powers, and Grenager (2007). We, too, believe that this is not where the analysis should end; rather, we think of it as a starting point for further investigations and future research, in order to identify the assumptions and conditions necessary to establish equilibrium.

[3] The problem of "truncated" observation of excess demand, hence the inability of producers to observe the true demand quantity given a certain price in case that demand is higher than actual supply, is present in reality of course. Since in this model, producers' price and supply quantity decisions directly depend on the consumer demand curve estimation solely relying on observed data, we are not able to relax this assumption without

distribute their spendable income among goods in a utility maximizing way, but they neither know the functional form of their utility function, nor have they any (direct) optimization abilities. Consequently, consumers have to rely on their experience of past consumption decisions, and they have learn how to behave optimally. We assume consumer income to be exogenously given. The consumer learning method is presented in detail in section 3.2.1.

The period is over once either all goods are sold out or all consumer income is spent; in the latter case, producer are not able to store their overproduction. By then, the next period starts with the producers price and supply quantity announcements. Each producer either keeps his price and supply strategy from the preceding period by announcing the same price and providing the same quantity of his good again, or, with some probability π_j, he revises his strategy and changes the price and the supply quantity. In the latter case, the producer chooses price and quantity such that his profits from expected sales are maximized; the estimation of expected sales given a certain price (the demand curve) will be discussed in detail in section 3.2.2. The total number of producers J is constant over time: producers can neither go bankrupt, nor are there market entries. Since each producer has its unique good, there are J different goods for consumption. However, the relationship between goods, meaning their (dis-)similarities, are modelled solely by consumers' preferences. None of the agents knows anything about the others' preferences or production technologies. The only channel of information between agents are prices, aggregated demand and supply quantities.

3.2.1 Consumer Learning

Let $\mathcal{J}^i_{t,n} := \{j^i_{t,k}\}_{k=1}^n$, $j \in \{1, \ldots, J\}$ be the sequence of goods consumed by agent i in period t, a total of n. Furthermore, let $u^i_{t,n} := U^i_{t,n} - U^i_{t,n-1}$ be the realized marginal utility of agent i in period t, where $U^i_{t,n}$ is the utility obtained from the consumption of $\mathcal{J}^i_{t,n}$. In words, $u^i_{t,n}$ is the increase of the overall utility level of one agent after having consumed one more good, namely good $j^i_{t,n}$. Then, the *propensity* of choosing good j for consumption next is

$$r^i_{t,n,j} = \begin{cases} r^i_{t,n-1,j}\,\bar{\gamma} + \alpha\,\dfrac{u^i_{t,n}}{p_{t,j}} & j^i_{t,n} = j \\ r^i_{t,n-1,j} & j^i_{t,n} \neq j \end{cases} \qquad 0 < \bar{\gamma} < 1. \tag{3.1}$$

which is the sum of the marginal utility obtained from consuming $\mathcal{J}^i_{t,n}$, per unit of money spend, scaled by parameter α; The factor $\bar{\gamma}$ implements forgetting within a period.[4] The probability of agent i choosing good j is derived as follows: Let $\tilde{J}^i_{t,n}$ be the set of goods that

introducing a significant estimation bias, which in turn would affect the overall market outcome.

[4]We assume that *within* a consumption period, forgetting is applied to the past consumption experience for the actually chosen good only; experiments with forgetting applied to all possible actions returned – once calibrated properly – similar results, but slightly more variance.

agent i can afford and that are still available on the market. Then,

$$pr^i_{t,n,j} := \begin{cases} r^i_{t,n,j} \cdot \Big(\sum\limits_{\substack{l=1 \\ l \in \tilde{J}^i_{t,n}}}^{J} r^i_{t,n,l} \Big)^{-1} & j \in \tilde{J}^i_{t,n} \\ 0 & j \notin \tilde{J}^i_{t,n} \end{cases} \tag{3.2}$$

Additionally, we set a lower bound on consumption choice probabilities $pr^i_{t,n,j} \geq \epsilon$ in order to ensure exploration of the whole action space.

It remains to define initial propensities at the beginning of each trading period, $r^i_{t,0,j}$. This is of great importance, since so far, consumers only maximize their utility within one period where goods prices stay fixed. However, if producers revise their price/quantity decisions at the beginning of a new period, consumers optimal response might change. In order to learn this, old experience has to be given up (forgotten); but in order keep consumers continuing as before in case prices don't change, initial propensities of period $t+1$ should be proportional to final choice probabilities at the end of period t. Consequently, we define

$$r^i_{t+1,0,j} = \tilde{\gamma} \cdot pr^i_{t,n,j} \tag{3.3}$$

where $\tilde{\gamma}$ controls the strength of initial propensities.

3.2.2 Producer Learning

Let $C^j(q^S)$ be the total cost function of producer j for output q^S. We assume that the cost function is known, and marginal costs MC^j (derivative of C^j with respect to q^S) can be derived. Consequently, profit maximization implies setting marginal costs equal to marginal revenues MR^j (derivative of total returns TR^j with respect to q^S).[5] Since the computation of MR^j requires the knowledge of consumer demand curve (demand quantities for all possible prices), which is unknown by assumption, producers apply the following regression learning procedure:

Define $\tilde{T}_{t,j}$ to be the moving time window of aggregated consumer demand observation of length T_j, $\tilde{T}_{t,j} := \big\{u \in \{t, t-1, \ldots, t-T_j\} \,|\, \mathbb{1}_{u-1} = 1\big\}$, where $\mathbb{1}_t$ is an index function taking on the value 1 if *any* producer changed its price at period t, and zero otherwise. Then, the linear demand function estimation model for time window $\tilde{T}_{t,j}$ and regression parameter vector β is

$$q^D_{\tilde{T}_{t,j},j} = \beta_0 + \beta_j p_{\tilde{T}_{t,j},j} + \beta_{-j} p_{\tilde{T}_{t,j},-j} + \varepsilon_j \tag{3.4}$$

where $q^D_{\tilde{T}_{t,j},j}$ is the history of demand observations for good j in $\tilde{T}_{t,j}$, and $p_{\tilde{T}_{t,j},\cdot}$ are the corresponding prices; Index $-j$ stands for the set of all goods except j, $-j = \{k : k \in \{1, \ldots, J\}, k \neq$

[5] This follows directly from the definition of total profits, which are equal to total returns less total cost, and setting its derivative to 0.

$j\}$. Consequently, the predicted demand for $t+1$ as a function of his and his competitors prices, $p_{t+1,j}$ and $\hat{p}_{t+1,-j}$, respectively, is

$$\hat{q}^D_{t+1,j} = \hat{\beta}_0 + \hat{\beta}_j p_{t+1,j} + \hat{\beta}_{-j}\hat{p}_{t+1,-j}. \tag{3.5}$$

Since the producers do not now the true future prices of their competitors $p_{t+1,-j}$, they will assume them to grow by their average growth rate within $\tilde{T}_{t,j}$. Since marginal costs and returns are in money rather than in quantity terms, we have to invert (3.5) to get the so called inverse demand function. Since at this point, the observed demand quantity $q^D_{\tilde{T}_{t,j},j}$, that was assumed to be a function of the price $p_{t+1,j}$, becomes in fact the decision variable dictating the price necessary to sell all units produced, we denote it as $q^S_{t+1,j}$ from now on. Finally, the inverse demand function writes as

$$p_{t+1,j} = \tilde{\beta}_0 + \tilde{\beta}_j q^S_{t+1,j}, \quad \tilde{\beta}_0 := -\frac{\hat{\beta}_0 + \hat{\beta}_{-j}\hat{p}_{t+1,-j}}{\hat{\beta}_j}, \quad \tilde{\beta}_j := \frac{1}{\hat{\beta}_j}. \tag{3.6}$$

Taking the derivative of total returns yields marginal returns as a function of the intended supply quantity,

$$MR^j(q^S) = \tilde{\beta}_0 + 2\tilde{\beta}_j q^S. \tag{3.7}$$

Finally, producers set prices and quantities such that $MC^j(q^S) = MR^j(q^S)$ and (3.6) is fulfilled. The full market event sequence is summarized in Algorithm 3.1.

3.3 Simulation Results and Validation

The presentation of the simulation results and their validation will be organized as follows: First, we validate the consumer learning procedure by fixing the price at some arbitrary level, and compare it to the benchmark. Then, we validate the producer regression learning procedure, independently of consumer learning, by letting consumers respond optimally as under perfect information. Last, we put things together and simulate and validate the whole model.

We parametrize the model as follows: There are two goods offered on the market, and consumers' utility function is $U(q_1, q_2) = 1.2 \cdot q_1^\sigma + q_2^\sigma$ with $\sigma = 0.7$ in order to make the goods non-perfect substitutes. For the consumer validation, prices and income s are fixed at $p_1 = 1.4$, $p_2 = 1$ and $s = 100$. Forgetting and step size are $\bar{\gamma} = .9999$, $\tilde{\gamma} = 100$, and $\alpha = 1$. Each simulation involves 100 agents and 100 consumption periods.

The simulation results for the consumer validation are reported in Fig. 3.1a, which depicts a histogram of the terminal relative deviation of the consumption quantity (good 1) from its benchmark solution.[6] We see that the distribution is unbiased, symmetrical, and almost all observations lie within 1.5 percent of deviation from the benchmark, which we think is very satisfying. For the simulations, we used three different values of initial propensities $r^i_{0,0,j}$, with 1000 runs each. The colour further dividing the histogram bars represent the share of

[6] Figures 3.1a is trimmed at the one percent level.

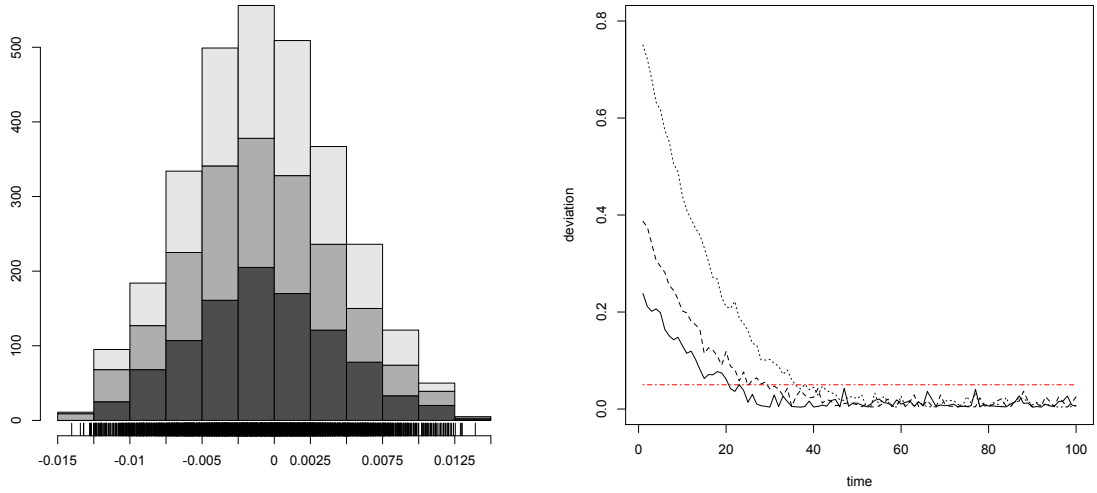
Algorithm 3.1 Artificial consumption goods market

initialize algorithm
for number of iterations (index t) **do**
    ## supply side:
    for all producers j **do**
        if j revises strategy **then**
            estimate demand function: (3.4)
            optimize profits: choose $p_{t+1,j}$ and $q^S_{t+1,j}$ s. t. $MR^j = MC^j$, (3.6)
        else
            set $q^S_{t+1,j} = q^S_{t,j}$ and $p_{t+1,j} = p_{t,j}$
        end if
    end for
    ## demand side:
    initialize propensities: (3.3)
    while (not all goods are sold out) and (not all income is spent) (index n) **do**
        for all consumers i **do**
            choose affordable good $j^i_{t+1,n}$ for consumption: (3.2)
            update propensities: (3.1)
        end for
    end while
    ## supply side:
    update demand history $q^D_{\tilde{T}_{t+1,j},j}$
end for

observations starting from each particular inital propensity value, and we conclude that they do not affect the final simulation outcome. Figure 3.1b shows the same deviation measure of three particular simulation runs (again starting from three different initial propensities) over time.

We now turn to the discussion of the producer learning algorithm. As indicated, we first run it based on the optimal consumer response, in order to validate this particular algorithm and rule out effects arising from the interoperation with the consumer learning. We assume the cost function to be quadratic and parametrized such that the minimum of average costs is at 50000, and the minimum of MC^j is at 32000. The observation time window length is $T_j = 2000$, and producers change their prices and quantities every 100 periods in average. There are 1500 consumers, each with 16000 units of income; to control for nominal effects of the much higher equilibrium prices compared to the consumer validation example, we set $\alpha = 400$. Figure 3.2a presents the results as a two-dimensional histogram of the joint terminal state distribution of the demand curve slope estimate (relative deviation from benchmark value)[7]; above and beside it, the marginal distributions are plotted, showing the impact of three different starting points

[7]Figure 3.2a is trimmed at the eight percent level.



(a) Terminal relative deviation from benchmark.

(b) Relative deviation from benchmark over time (slash-dotted: 5% line).

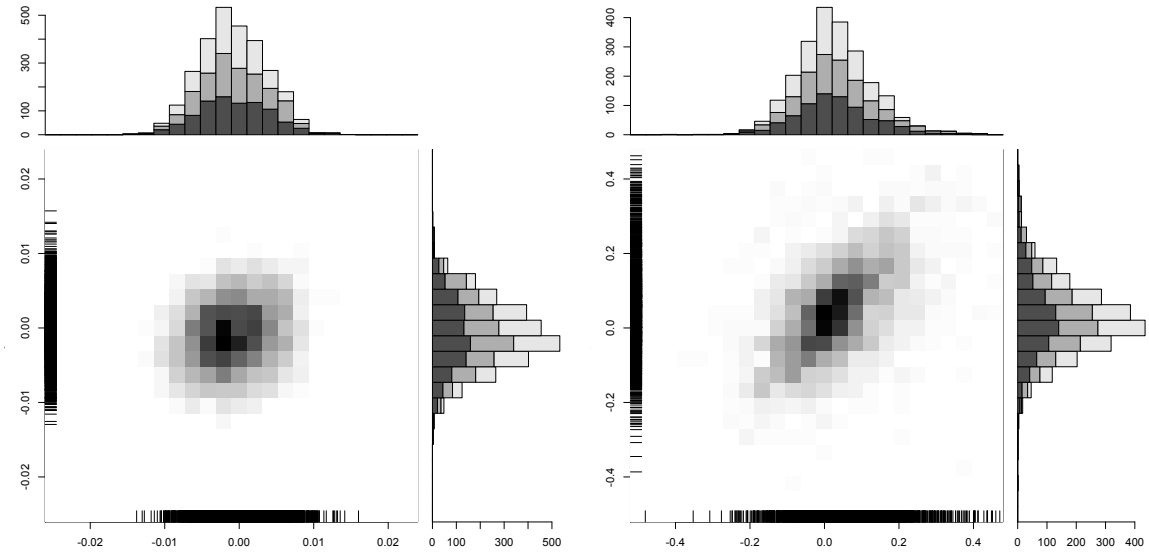
Figure 3.1: Simulation and validation results for the consumer learning algorithm for three values of initial propensities.

for the estimation, similar to what we did in Fig. 3.1a. For each of them, 1000 simulations of 25000 periods length were carried out. We conclude that the algorithm is nearly unbiased, starting point independent, and that it has very low variance. Additionally, the distribution seems to have no correlation, so the direction of deviation from the benchmark of one producer is independent of the other one's; hence, we think of it as being noise only.

Turning to the full market simulation, we now simulate both the consumer and the producer as learning agents; the results are presented in Fig. 3.2b. Again, the market seems to converge towards equilibrium (unbiased), but this time, the variance of the terminal states of the system is much higher.[8] Moreover, there is significant correlation in demand curve estimation error. We think that this is due to the fact that learning consumers take longer to adapt to changes in consumption good prices, and hence producers take longer to learn that it is optimal for them to be closer to the benchmark than their competitors.

However, the convergence of the full market simulation towards the equilibrium comes at no surprise: The configuration of utility and cost functions we use is such that a unique Nash equilibrium is established. Moreover, this Nash equilibrium is attractive. In the production context, this means that (i) in equilibrium, no producer can increase its profits given the other producers strategies and therefore all players remain passive, and (ii) out of equilibrium, for the producer with the highest deviation from the benchmark (in terms of profits), it is always

[8] About 25% of the simulation runs of the full market simulation ended early, because the system got destabilized and moved towards states where demand curve estimation was impossible. These observations are not included in Fig. 3.2b.



(a) Optimal consumer response.

(b) Consumer learning response.

Figure 3.2: Terminal relative deviation from benchmark for the producer regression learning algorithm for three values of initial propensities.

more profitable to correct his strategy towards it, and no producer can increase his profits by moving further away from the benchmark than the producer with the highest deviation. Consequently, if consumer learning for fixed prices yields nearly optimal consumption bundles, and if producer approximate the true demand curve correctly for optimal consumer response (as it is the case for the utility function in use), the market will finally converge. Of course, further research is needed for utility configurations with multiple or lacking Nash equilibria, possibly revealing periodic or even chaotic behaviour.

Checking our model for sensitivity to changes in parameter values, we find that with respect to the producer learning, the model seems to be robust as long as (i) the frequency of producers changing prices and output quantities π_j, in relation to the demand observation window length T_J, provides high enough a number of observations for the estimation procedure, and (ii) the consumer agents have enough time to adapt to the new price regime and therefore respond in a way that reflects their actual preferences. For the consumer learning algorithm, forgetting (in relation to the step size parameter α and the coefficients of the underlying utility function) needs to be calibrated properly. We find that our specific setting is robust with respect to the ratio of goods prices, as long as α is in the order of magnitude of the prices.

3.4 Conclusions and Outlook

We have presented an Agent-based model of a market for heterogeneous consumption goods, where consumers do not know their utility function, but can only learn to maximize it from

past consumption experience; at the same time, producers estimate the demand curve for their goods from historical data in order to maximize their profits. Our simulation results indicate that the market is self-organizing, and moreover, establishes an optimal state from the viewpoint of profit and utility maximization, without the assumption of perfect information about agents' preferences and technologies, and without a central authority that organizes the market. However, relating our result to reality, we have to conclude that even if the model would be correct, and reality would be just one draw from the distribution in Fig. 3.2b, variance is still to high to conclude that the real world is likely to be at its profit and utility maximizing equilibrium.

Further research will on the one hand investigate the market's convergence (or periodicity) behaviour for different utility functions, since we expect the existence and uniqueness of the Nash equilibrium of the current configuration to be the driving force behind our results. On the other hand, we will ask whether the market is also stable, and moreover, whether the "law of one price" is established in case that the goods are perfect substitutes, meaning that they are perfectly interchangeable for agents without any loss in utility. In the existing literature, this can only be achieved by assuming that goods prices are no common knowledge among agents, an assumption we think to be unrealistic, too.

Part III

Bibliography and Curriculum Vitae

Bibliography

Aguirregabiria, V. and P. Mira (2010): "Dynamic Discrete Choice Structural Models: A Survey," *Journal of Econometrics*, 156, 38–67.

Akima, H. (1970): "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures," *Journal of the ACM*, 17, 589–602.

Arcidiacono, P. and P. B. Ellickson (2011): "Practical Methods for Estimation of Dynamic Discrete Choice Models," *Annual Review of Economics*, 3, 363–394.

Arcidiacono, P. and R. A. Miller (2011): "Conditional Choice Probability Estimation of Dynamic Discrete Choice Models with Unobserved Heterogeneity," *Econometrica: Journal of the Econometric Society*, 79, 1823–1867.

Baines, M. J. (1998): "Grid Adaptation via Node Movement," *Applied Numerical Mathematics*, 26, 77–96.

Bates, D., K. M. Mullen, J. C. Nash, and R. Varadhan (2012): *minqa: Derivative-Free Optimization Algorithms by Quadratic Approximation*, R package version 1.2.3.

Bellman, R. (1952): "On the Theory of Dynamic Programming," *Proceedings of the National Academy of Sciences of the United States of America*, 38, 716–719.

Blevins, J. R. (2011): "Sequential Monte Carlo Methods for Estimating Dynamic Microeconomic Models," Tech. rep., Ohio State University.

Blocker, A. W. (2011): *fastGHQuad: Fast Rcpp Implementation of Gauss–Hermite Quadrature*, R package version 0.1-1.

Borchers, H. W. (2014): *pracma: Practical Numerical Math Functions*, R package version 1.6.4.

Bowling, M. and M. Veloso (2002): "Multiagent Learning Using a Variable Learning Rate," *Artificial Intelligence*, 136, 215–250.

Brumm, J. and S. Scheidegger (2013): "Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models," Tech. rep., University of Zurich.

Busoniu, L., R. Babuska, and B. De Schutter (2008): "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, 38, 156–172.

Cai, Y. and K. L. Judd (2013): "Advances in Numerical Dynamic Programming and New Applications," in *Handbook of Computational Economics*, ed. by K. Schmedders and K. L. Judd, Elsevier, 479–516.

Ciarlet, P. G. (2002): *The Finite Element Method for Elliptic Problems*, SIAM.

Colson, B., P. Marcotte, and G. Savard (2007): "An Overview of Bilevel Optimization," *Annals of Operations Research*, 153, 235–256.

Cosslett, S. R. and L.-F. Lee (1985): "Serial Correlation in Latent Discrete Variable Models," *Journal of Econometrics*, 27, 79–97.

de Boor, C. (2001): *A Practical Guide to Splines*, Springer.

Eddelbuettel, D. and R. François (2011): "Rcpp: Seamless R and C++ Integration," *Journal of Statistical Software*, 40, 1–18.

Erev, I. and A. E. Roth (1998): "Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria," *American Economic Review*, 88, 848–881.

Fletcher, R., S. Leyffer, D. Ralph, and S. Scholtes (2006): "Local Convergence of SQP Methods for Mathematical Programs with Equilibrium Constraints," *SIAM Journal on Optimization*, 17, 259–286.

Fraser, W. (1965): "A Survey of Methods of Computing Minimax and Near-Minimax Polynomial Approximations for Functions of a Single Independent Variable," *Journal of the ACM*, 12, 295–314.

Galassi, M., J. Davies, J. Theiler, B. Gough, G. Jungman, P. Alken, M. Booth, and F. Rossi (2014): *GNU Scientific Library Reference Manual*, version 1.16.

Gintis, H. (2007): "The Dynamics of General Equilibrium," *The Economic Journal*, 117, 1280–1309.

——— (2011): "The Dynamics of Generalized Market Exchange," Tech. rep., Santa Fe Institute.

Grüne, L. (1997): "An Adaptive Grid Scheme for the Discrete Hamilton-Jacobi-Bellman Equation," *Numerische Mathematik*, 75, 319–337.

Grüne, L. and W. Semmler (2004): "Using Dynamic Programming with Adaptive Grid Scheme for Optimal Control Problems in Economics," *Journal of Economic Dynamics and Control*, 28, 2427–2456.

Hasselman, B. (2014): *nleqslv: Solve Systems of Non-Linear Equations*, R package version 2.1.1.

Hotz, V. J. and R. A. Miller (1993): "Conditional Choice Probabilities and the Estimation of Dynamic Models," *The Review of Economic Studies*, 60, 497.

Hsieh, C.-Y. and S. L. Mangum (1986): *A Search for Synthesis in Economic Theory*, M.E. Sharpe.

Huang, W. and R. D. Russell (2011): *Adaptive Moving Mesh Methods*, Springer.

Imai, S., N. Jain, and A. Ching (2009): "Bayesian Estimation of Dynamic Discrete Choice Models," *Econometrica: Journal of the Econometric Society*, 77, 1865–1899.

Jäckel, P. (2005): "A Note on Multivariate Gauss-Hermite Quadrature," Tech. rep.

Judd, K. L. (1992): "Projection Methods for Solving Aggregate Growth Models," *Journal of Economic Theory*, 58, 410–452.

——— (1998): *Numerical Methods in Economics*, The MIT Press.

Kay, S. M. (1983): "Recursive Maximum Likelihood Estimation of Autoregressive Processes," *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31, 56–65.

Keane, M. P., P. E. Todd, and K. I. Wolpin (2011): "The Structural Estimation of Behavioral Models: Discrete Choice Dynamic Programming Methods and Applications," in *Handbook of Labor Economics*, ed. by O. Ashenfelter and D. Card, Elsevier, 331–461.

Keane, M. P. and K. I. Wolpin (1994): "The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation and Interpolation: Monte Carlo Evidence," *The Review of Economics and Statistics*, 76, 648–672.

Kristensen, D. and B. Schjerning (2014): "Implementation and Estimation of Discrete Markov Decision Models by Sieve Approximation," Tech. rep.

Kythe, P. K. and M. R. Schäferkotter (2005): *Handbook of Computational Methods for Integration*, vol. 1, CRC Press.

Larsen, B. J., F. Oswald, G. Reich, and D. Wunderli (2012): "A Test of the Extreme Value Type I Assumption in the Bus Engine Replacement Model," *Economics Letters*, 116, 213–216.

Lawson, C. L. (1964): "Characteristic Properties of the Segmented Rational Minimax Approximation Problem," *Numerische Mathematik*, 6, 293–301.

McFadden, D. (1974): "Conditional Logit Analysis of Qualitative Choice Behavior," in *Frontiers in Econometrics*, ed. by P. Zarembka, Academic Press, 105–142.

——— (1981): "Econometric Models for Probabilistic Choice," in *Structural Analysis of Discrete Data with Econometric Applications*, ed. by C. F. Manski and D. McFadden, The MIT Press, 198–272.

Norets, A. (2009): "Inference in Dynamic Discrete Choice Models with Serially Correlated Unobserved State Variables," *Econometrica: Journal of the Econometric Society*, 77, 1665–1682.

——— (2012): "Estimation of Dynamic Discrete Choice Models Using Artificial Neural Network Approximations," *Econometric Reviews*, 31, 84–106.

Powell, M. J. D. (2009): "The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives," Tech. rep., University of Cambridge.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007): *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, 3 ed.

R Core Team (2014): *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing.

Reich, G. (2011): "Market Self-organization under Limited Information," in *Advances in Artificial Intelligence*, Springer, 32–41.

——— (2014): "Divide and Conquer: Recursive Likelihood Function Integration for Dynamic Discrete Choice Models with Serially Correlated Unobserved State Variables," *Submitted to Operations Research*.

Rieskamp, J., J. R. Busemeyer, and T. Laine (2003): "How Do People Learn to Allocate Resources? Comparing Two Learning Theories." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 29, 1066–1081.

Rothschild, C. (2010): "Sins of the Sons of Samuelson: Vision, Pedagogy, and the Zig-Zag Windings of Complex Dynamics," *Journal of Economic Behavior & Organization*, 74, 277–290.

Rouchier, J. (2008): "Agent-Based Simulation as a Useful Tool for the Study of Markets," Tech. rep., GREQAM.

Rust, J. (1987): "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher," *Econometrica: Journal of the Econometric Society*, 55, 999–1033.

——— (1988): "Maximum Likelihood Estimation of Discrete Control Processes," *SIAM Journal on Control and Optimization*, 26, 1006–1024.

——— (1996): "Numerical Dynamic Programming in Economics," in *Handbook of Computational Economics*, ed. by H. M. Amman, D. A. Kendrick, and J. Rust, Elsevier, 619–729.

Schenk, O. and K. Gärtner (2004): "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO," *Future Generation Computer Systems*, 20, 475–487.

Schumaker, L. (1968): "Uniform Approximation by Chebyshev Spline Functions. II: Free Knots," *SIAM Journal on Numerical Analysis*, 5, 647–656.

——— (2007): *Spline Functions: Basic Theory*, Cambridge University Press.

Shoham, Y., R. Powers, and T. Grenager (2007): "If Multi-Agent Learning is the Answer, what is the Question?" *Artificial Intelligence*, 171, 365–377.

Spall, J. C. (2003): *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, Wiley-Interscience.

Stephenson, A. G. (2002): "evd: Extreme Value Distributions," *R News*, 2.

Stinebrickner, T. R. (2000): "Serially Correlated Variables in Dynamic, Discrete Choice Models," *Journal of Applied Econometrics*, 15, 595–624.

Su, C.-L. and K. L. Judd (2012): "Constrained Optimization Approaches to Estimation of Structural Models," *Econometrica: Journal of the Econometric Society*, 80, 2213–2230.

Tesfatsion, L. and K. L. Judd (2006): *Agent-Based Computational Economics*, vol. 2 of *Handbook of Computational Economics*, Elsevier.

Thompson, J. F., B. K. Soni, and N. P. Weatherill (2010): *Handbook of Grid Generation*, CRC Press.

Vriend, N. J. (2001): "Evolving Market Structure: An ACE Model of Price Dispersion and Loyalty," *Journal of Economic Dynamics and Control*, 25, 459–502.

Wächter, A. and L. T. Biegler (2005): "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, 106, 25–57.

Walras, L. (1954): *Elements of Pure Economics, or, The Theory of Social Wealth*, Allen and Unwin.

Wang, X. F. and T. Sandholm (2003): "Reinforcement Learning to Play an Optimal Nash Equilibrium in Team Markov Games," *Advances in neural information processing systems*, 1603–1610.

Weisbuch, G. and D. Herreiner (2000): "Market Organisation and Trading Relationships," *The Economic Journal*, 110, 411–436.